LAPORAN TUGAS BESAR
IF2211 STRATEGI ALGORITMA

PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PERMAINAN DIAMONDS

Dipersiapkan oleh:

Diwan Ramadhani Dwi Putra	123140116
M. Gymnastiar Syahputra	123140135
Jordy Anugrah Akbar	123140141

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK INDUSTRI

INSTITUT TEKNOLOGI SUMATERA

2025

DAFTAR ISI

DAFTAR GAMBAR

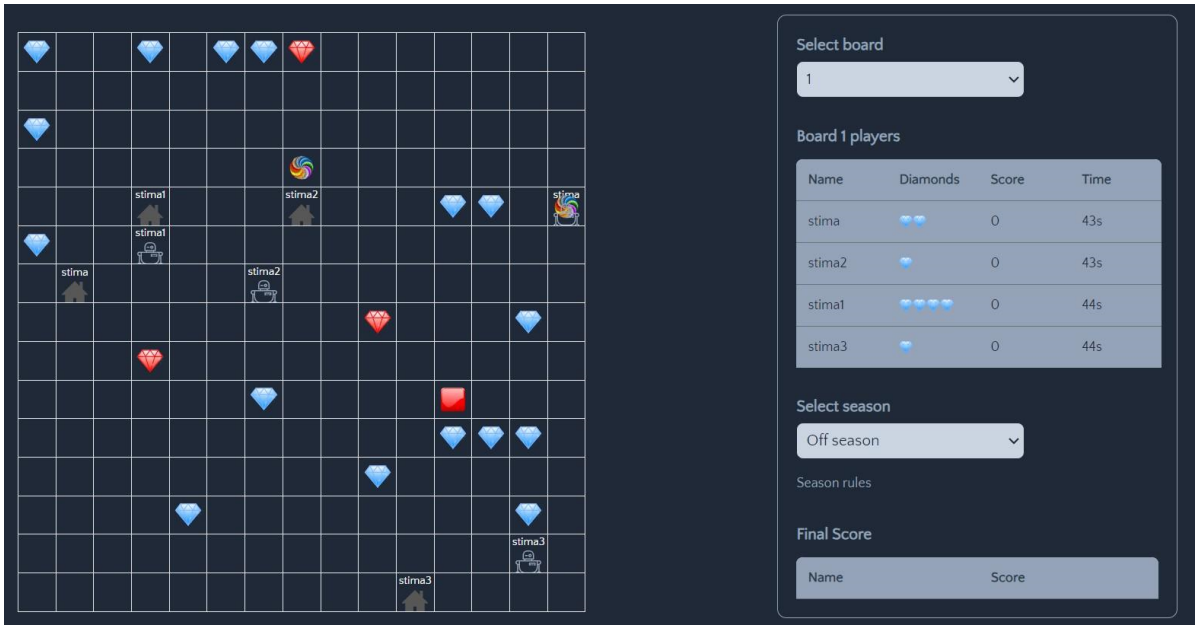
DAFTAR TABEL

DAFTAR LAMPIRAN

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 sesederhana 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.



Gambar 1. 1 Tampilan Awal Permainan

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.

Program permainan Diamonds terdiri atas:

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

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

Komponen-komponen dari permainan Diamonds antara lain:

1. **Diamonds**

Gambar 1. 2 Varian Diamond Biru

Gambar 1. 3 Varian Diamond Merah

Untuk memenangkan pertandingan, kita harus mengumpulkan *diamond* ini sebanyak-banyaknya dengan melewati/melangkahinya. 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**

Gambar 1. 4 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**

Gambar 1. 5 Teleporter

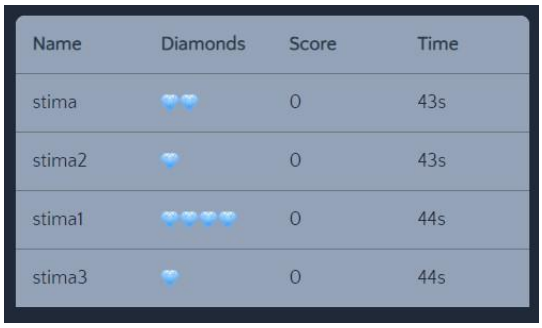
Terdapat 2 *teleporter* yang saling terhubung satu sama lain. Jika bot melewati sebuah *teleporter* maka bot akan berpindah menuju posisi *teleporter* yang lain.

4. **Bots and Bases**

Gambar 1. 6 Bot dan Basenya

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.

5. **Inventory**



Gambar 1. 7 Total Diamond yang dimiliki Bot

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.

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

1. Pertama, setiap pemain (bot) akan ditempatkan pada *board* secara *random*. Masing-masing bot akan mempunyai *home base*, serta memiliki *score* dan *inventory* awal bernilai nol.
2. Setiap bot diberikan waktu untuk bergerak, waktu yang diberikan semua sama untuk setiap pemain.
3. Objektif utama bot adalah mengambil *diamond-diamond* yang ada di peta sebanyak-banyaknya. Seperti yang sudah disebutkan di atas, *diamond* yang berwarna merah memiliki 2 poin dan *diamond* yang berwarna biru memiliki 1 poin.
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 menimpa 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.

Permainan ini merupakan permainan berbasis *web*, sehingga setiap aksi yang dilakukan – mulai dari mendaftarkan bot hingga menjalankan aksi bot – akan memerlukan HTTP *request* terhadap API *endpoint* tertentu yang disediakan oleh *backend*. Berikut adalah urutan *requests* yang terjadi dari awal mula permainan.

1. Program bot akan mengecek apakah bot sudah terdaftar atau belum, dengan mengirimkan *POST request* terhadap *endpoint* `/api/bots/recover` dengan *body* berisi *email* dan *password* bot. Jika bot sudah terdaftar, maka *backend* akan memberikan *response code* 200 dengan *body* berisi id dari bot tersebut. Jika tidak, *backend* akan memberikan *response code* 404.
2. Jika bot belum terdaftar, maka program bot akan mengirimkan *POST request* terhadap *endpoint* `/api/bots` dengan *body* berisi *email, name, password*, dan *team*. Jika berhasil, maka *backend* akan memberikan *response code* 200 dengan *body* berisi id dari bot tersebut.
3. Ketika id bot sudah diketahui, bot dapat bergabung ke *board* dengan mengirimkan *POST request* terhadap *endpoint* `/api/bots/{id}/join` dengan *body* berisi board id yang diinginkan

(preferredBoardId). Apabila bot berhasil bergabung, maka *backend* akan memberikan *response code* 200 dengan *body* berisi informasi dari *board*.

4. Program bot akan mengkalkulasikan *move* selanjutnya secara berkala berdasarkan kondisi *board* yang diketahui, dan mengirimkan *POST request* terhadap *endpoint* /api/bots/{id}/move dengan *body* berisi *direction* yang akan ditempuh selanjutnya ("NORTH", "SOUTH", "EAST", atau "WEST"). Apabila berhasil, maka *backend* akan memberikan *response code* 200 dengan *body* berisi kondisi *board* setelah *move* tersebut. Langkah ini dilakukan terus-menerus hingga waktu bot habis. Jika waktu bot habis, bot secara otomatis akan dikeluarkan dari *board*.
5. Program *frontend* secara periodik juga akan mengirimkan *GET request* terhadap *endpoint* /api/boards/{id} untuk mendapatkan kondisi *board* terbaru, sehingga tampilan *board* pada *frontend* akan selalu ter-*update*.

BAB II
LANDASAN TEORI

A. Algortima Greedy

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

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

2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Dalam menjalankan algoritma greedy ada beberapa elemen yang diperlukan agar proses pemecahan masalah secara greedy lebih mudah untuk dilakukan. Elemen- elemen tersebut adalah sebagai berikut.

1. Himpunan kandidat (C), berisi kandidat yang akan dipilih pada setiap Langkah. Contohnya adalah simpul atau sisi di dalam graf, job, task, koin, benda, dan karakter.

2. Himpunan solusi (S), berisi kandidat yang sudah dipilih untuk menjadi solusi.

3. Fungsi solusi yang digunakan menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi atau belum.

4. Fungsi seleksi (selection function) untuk memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik yang tentunya berbeda untuk setiap masalah yang dihadapi.

5. Fungsi kelayakan (feasible) untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi atau tidak.

6. Fungsi objektif untuk memaksimumkan atau meminimumkan kandidat yang dipilih.

Setelah semua elemen- elemen tersebut lengkap, maka proses pencarian solusi akan bisa dilakukan. Algoritma greedy melibatkan pencarian sebuah himpunan bagian (S) dari himpunan kandidat (C) yang dalam hal ini, himpunan S harus memenuhi beberapa kriteria 7 yang ditentukan, yaitu S menyatakan suatu solusi dan S di optimisasi oleh fungsi objektif. Algoritma greedy akan menghasilkan beberapa solusi optimum lokal dan dari semua solusi lokal tersebut akan dipilih solusi terbaik yang menjadi solusi optimum global.

Akan tetapi, solusi optimum global yang dihasilkan algoritma greedy belum tentu merupakan solusi terbaik karena sangat mungkin solusi tersebut merupakan solusi sub-optimum atau pseudo-optimum. Ada dua alasan kenapa hal tersebut bisa terjadi, yaitu sebagai berikut:

1. Algoritma greedy tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada (sebagaimana pada metode exhaustive search).

2. Terdapat beberapa fungsi seleksi yang berbeda, sehingga perlu memilih fungsi yang tepat jika algoritma diinginkan untuk menghasilkan solusi optimal.

Sebagai kesimpulan, algoritma greedy adalah metode penyelesaian masalah secara bertahap dengan memilih solusi terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang, dengan harapan solusi lokal yang dipilih akan menghasilkan solusi global yang optimal. Untuk menjalankan algoritma ini secara efektif, diperlukan beberapa elemen penting seperti himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif. Meskipun efisien, algoritma greedy tidak selalu menjamin solusi optimal secara global karena tidak mengevaluasi semua kemungkinan solusi dan sangat bergantung pada strategi seleksi yang digunakan. Oleh karena itu, pemilihan fungsi seleksi yang tepat sangat penting untuk mendekati hasil yang optimal.

Dalam penerapannya, ada cukup banyak permasalahan yang bisa diselesaikan dengan pendekatan greedy, yaitu sebagai berikut:

1. Persoalan penukaran uang (coin exchange problem)

2. Persoalan memilih aktivitas (activity selection problem)

3. Minimisasi waktu di dalam system

4. Persoalan knapsack (knapsack problem)

5. Penjadwalan job dengan tenggat waktu (job schedulling with deadlines)

6. Pohon merentang minimum (minimum spanning tree)

7. Lintasan terpendek (shortest path)

8. Kode Huffman (Huffman code)

9. Pecahan Mesir (Egyptian fraction)

B. Cara Kerja Program

1. Struktur Direktori Program

Program Bot yang dibuat dalam tugas besar 1 memiliki struktur direktori sebagai berikut:

```
Root Directory
└── tubes1-IF2211-bot-starter-pack-1.0.1
    ├── __pycache__/
    ├── game/
    │   ├── __pycache__/
    │   ├── logic/
    │   │   ├── __pycache__/
    │   │   ├── __init__.py
    │   │   ├── base.py
    │   │   ├── cost_efficient.py
    │   │   ├── risk_aware.py
    │   │   ├── random.py
    │   │   ├── time_weighted.py
    │   │   ├── valueToDistance.py
    │   ├── __init__.py
    │   ├── api.py
    │   ├── board_handler.py
    │   ├── bot_handler.py
    │   ├── models.py
    │   └── util.py
    ├── .gitignore
    ├── decode.py
    ├── main.py
    ├── README.md
    ├── requirements.txt
    ├── run-bots.bat
    └── run-bots.sh
```

2. Proses Pelaksanaan Aksi oleh Bot

Permainan dalam tugas besar ini merupakan permainan berbasis web, sehingga setiap aksi yang dilakukan– mulai dari mendaftarkan bot hingga menjalankan aksi bot akan memerlukan HTTP request terhadap API endpoint tertentu yang disediakan oleh backend. Berikut adalah urutan requests yang terjadi dari awal mula permainan.

a. Program bot akan melakukan pengecekan apakah bot sudah terdaftar atau belum dengan mengirimkan POST request terhadap endpoint /api/bots/recover dengan body berisi email dan password bot. Jika bot sudah terdaftar, maka backend akan

memberikan response code 200 dengan body berisi id dari bot tersebut. Jika tidak, backend akan memberikan response code 404.

b. Jika bot belum terdaftar, maka program bot akan mengirimkan POST request terhadap endpoint /api/bots dengan body berisi email, name, password, dan team. Jika berhasil, maka backend akan memberikan response code 200 dengan body berisi id dari bot tersebut.
c. Ketika id bot sudah diketahui, bot dapat bergabung ke board dengan mengirimkan POST request terhadap endpoint /api/bots/{id}/join dengan body berisi board id yang diinginkan (preferredBoardId). Apabila bot berhasil bergabung, maka backend akan memberikan response code 200 dengan body berisi informasi dari board.
d. Program bot akan mengkalkulasikan move selanjutnya secara berkala berdasarkan kondisi board yang diketahui, dan mengirimkan POST request terhadap endpoint /api/bots/{id}/move dengan body berisi direction yang akan ditempuh selanjutnya ("NORTH", "SOUTH", "EAST", atau "WEST"). Apabila berhasil, maka backend akan memberikan response code 200 dengan body berisi kondisi board setelah move tersebut. Langkah ini dilakukan terus-menerus hingga waktu bot habis. Jika waktu bot habis, bot secara otomatis akan dikeluarkan dari board.
e. Program frontend secara periodik juga akan mengirimkan GET request terhadap endpoint /api/boards/{id} untuk mendapatkan kondisi board terbaru, sehingga tampilan board pada frontend akan selalu ter-update.

3. Implementasi Algoritma Greedy ke Dalam Bot

Untuk mengimplementasikan algoritma yang telah dibuat ke dalam bot, perlu dibuat terlebih dahulu sebuah folder baru pada direktori /game/logic, misalnya MyBot.py. Lalu Selanjut akan kelas baru yang meng-inherit kelas BaseLogic, lalu implementasikan constructor dan method next_move pada kelas tersebut. Berikut adalah contoh dari pembuatan kelas.

```
from game.logic.base import BaseLogic
from game.models import Board, GameObject

class MyBot(BaseLogic):
   def _init_(self):
          # Initialize attributes necessary
          self.my_attribute = 0
   def next_moveself, board_bot: GameObject, board: Board):
          # Calculate next move
```

```
        delta_x = 1
        delta_y = 0

        return delta_x, delta_y
```

Fungsi next_move pada kutipan di atas akan mengembalikan nilai delta_x dan delta_y, di mana nilai yang diperbolehkan hanyalah (1, 0), (0, 1), (-1, 0), (0,-1). Apabila nilai ilegal atau di-luar range board, maka move akan diabaikan oleh program dan akan mucul error Invalid Move.

Langkah selanjutnya adalah melakukan Import kelas yang telah dibuat pada main.py dan mendaftarkannya pada dictionary CONTROLLERS. Berikut adalah contoh dari proses Import kelas.

```
from game.logic.mybot import MyBot
init()
BASE_URL = "http://localhost:3000/api"
DEFAULT_BOARD_ID = 1
CONTROLLERS = {"Random": RandomLogic, "MyBot": MyBot}
```

Bot yang telah dibuat dapat dijalankan satu per satu atau dapat dijalankan secara bersamaan dengan bot lain dengan menggunakan .bat atau .sh script. Proses menjalankan program akan dijelaskan lebih lanjut pada poin 4

4. Menjalankan Program Bot

Program bot yang telah dibuat dapat dijalankan dengan menggunakan aplikasi cmd atau terminal. Untuk menjalankan satu bot dengan logic yang terdapat pada file game/logic/random.py diperlukan perintah seperti di bawah ini. Argumen logic pada command/script tersebut perlu disesuaikan menjadi nama bot yang telah terdaftar pada CONTROLLERS.

```
python main.py--logic Random--email=your_email@example.com--
name=your_name--password=your_password--team etimo
```

Untuk menjalankan beberapa bot sekaligus, misalnya menjalankan 4 bot dengan logic yang sama, yaitu game/logic/random.py. Perlu di buat terlebih dahulu file run-bots.bat pada Windows atau run-bots.sh pada Linux dan MacOS. File tersebut akan berisi perintah-perintah yaitu sebagai berikut.

```
@echo off
start cmd /c “python main.py--logic Random--email=your_email@example.com-
-name=your_name--password=your_password--team etimo”
```

```
start cmd /c "python main.py--logic Random--email=your_email@example.com--name=your_name--password=your_password--team etimo"
start cmd /c "python main.py--logic Random--email=your_email@example.com--name=your_name--password=your_password--team etimo"
start cmd /c "python main.py--logic Random--email=your_email@example.com--name=your_name--password=your_password--team etimo"
```

Script yang ada pada run-bots.bat atau run-bots.sh dapat disesuaikan sesuai dengan logic file yang digunakan, email, nama, ataupun password

BAB III

APLIKASI STRATEGI GREEDY

A. Mapping Persoalan Diamonds

Berdasarkan studi pustaka yang telah dilakukan pada landasan teori, dikatakan algoritma greedy melibatkan pencarian sebuah himpunan bagian (S) dari himpunan kandidat (C) yang dalam hal ini, himpunan S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S di optimisasi oleh fungsi objektif. Oleh karena itu, untuk bisa menerapkan algoritma greedy dalam permasalahan ini, maka perlu diidentifikasi terlebih dahulu elemen- elemen yang ada di dalam game Diamonds menjadi elemen- elemen algoritma greedy, yaitu sebagai berikut:

1. Himpunan kandidat (C)

   Himpunan kandidat dalam permasalahan ini akan berisi koordinat- koordinat objek yang ada di dalam game. Objek- objek tersebut adalah blue diamond, red diamond, red button, teleporter, base bot, bot musuh, dan base musuh. Selain itu, ada juga elemen penting yang perlu diperhatikan dalam game, yaitu jarak bot ke koordinat tujuan.

2. Himpunan solusi (S)

   Himpunan solusi berisi koordinat yang akan memberikan keuntungan paling banyak ketika didatangi oleh bot. Berdasarkan pendekatan greedy yang dilakukan, himpunan solusi bisa hanya berisi satu koordinat atau beberapa koordinat yang harus dilalui secara berurutan.

3. Fungsi solusi

   Fungsi solusi berguna untuk menentukan apakah kandidat yang dipilih sudah memberikan solusi. Secara umum, fungsi ini hanya akan digunakan ketika permainan akan berakhir. Fungsi ini akan melakukan perhitungan apakah waktu yang tersisa cukup untuk menempuh suatu koordinat dalam map dan untuk kembali ke base. Alasan kenapa fungsi ini hanya dilakukan di akhir permainan adalah karena fungsi-fungsi untuk menentukan kandidat solusi pada fungsi seleksi sudah menghilangkan kandidat yang tidak bisa menjadi solusi.

4. Fungsi Seleksi (Selection)

   Fungsi seleksi berguna untuk memilih kandidat berdasarkan strategi greedy tertentu. Fungsi seleksi akan dijelaskan lebih lanjut pada bagian alternatif solusi karena pendekatan greedy yang berbeda akan memiliki fungsi seleksi yang berbeda.

5. Fungsi Kelayakan (Feasible)

   Fungsi kelayakan berguna untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi. Dalam game Diamonds, fungsi kelayakan akan digunakan untuk menentukan apakah objek dalam game bisa menjadi tujuan berikutnya atau

tidak. Contoh yang paling sederhana adalah ketika red diamond menjadi tujuan selanjutnya, namun bot sudah memiliki 4 diamonds dalam inventory sehingga red diamond tidak bisa diambil. Fungsi kelayakan akan mencegah bot untuk menuju koordinat tersebut, karena apabila bot tetap menuju koordinat tersebut ada kemungkinan terjadi bug yaitu bot akan berputar- putar di sekitar red diamond atau bahkan memberikan invalid move.

6. Fungsi objektif

Fungsi objektif digunakan untuk memaksimalkan atau meminimumkan dari himpunan kandidat menjadi solusi. Dalam permasalahan ini himpunan kandidat akan melalui fungsi objektif untuk ditentukan mana kandidat yang memiliki keuntungan terbesar. Jadi, fungsi objektif akan diterapkan untuk memaksimalkan keuntungan yang didapat dari himpunan kandidat

B. Alternatif Solusi

1. Algoritma Greedy Cost-Efficient Path

Strategi ini adalah pendekatan greedy berbasis kondisi untuk mengumpulkan berlian seefisien mungkin dan kembali ke markas (base) saat diperlukan. Bot memilih target berdasarkan kedekatan (jarak efektif) dan nilai (poin) berlian, serta mempertimbangkan rute optimal termasuk penggunaan portal dan menghindari hambatan.

a. Himpunan Kandidat (Candidate Set)

Di setiap langkah permainan, bot menganalisis lingkungan untuk mengidentifikasi semua objek penting sebagai kandidat tindakan. Kandidat ini mencakup seluruh berlian yang tersedia di papan, portal teleportasi yang bisa digunakan untuk mempercepat perjalanan, serta tombol khusus seperti tombol merah. Semua objek ini menjadi pertimbangan awal dalam menentukan rute dan tujuan berikutnya.

b. Himpunan Solusi (Solution Set)

Dari himpunan kandidat, bot menentukan satu atau beberapa target yang akan menjadi jalur solusi. Ini bisa berupa satu titik tujuan langsung, seperti berlian terdekat, atau jalur kompleks yang terdiri dari beberapa titik — misalnya melalui portal sebelum menuju berlian atau base. Solusi ini dapat bersifat dinamis, berubah tergantung keadaan, dan disimpan dalam bentuk urutan posisi yang harus dicapai.

c. Fungsi Seleksi (Selection Function)

Bot memilih target dengan mempertimbangkan berbagai opsi dan membandingkan nilai efektifnya. Pemilihan dilakukan berdasarkan kombinasi antara jarak dan nilai poin dari objek, seperti memilih berlian dengan rasio

jarak/poin terkecil. Jika menggunakan portal lebih menguntungkan dibanding jalan langsung, bot akan memilih rute tersebut. Dengan demikian, pilihan selalu mengarah pada opsi yang paling efisien dalam konteks waktu dan skor.

d. Fungsi Kelayakan (Feasibility Function)

Setelah target dipilih, bot mengevaluasi kelayakan jalur menuju target. Ia memeriksa apakah ada hambatan seperti portal yang harus dilewati atau berlian merah yang harus dihindari. Jika rute langsung terhalang, bot merancang jalur alternatif dengan tujuan antara agar bisa menghindari rintangan sebelum kembali ke jalur utama. Ini memastikan bahwa jalur yang dipilih selalu bisa dieksekusi dengan aman.

e. Fungsi Solusi (Solution Function)

Strategi dibentuk melalui fungsi utama yang bertugas menentukan langkah selanjutnya berdasarkan kondisi permainan. Fungsi ini mempertimbangkan status berlian yang dimiliki, waktu yang tersisa, serta posisi objek penting. Berdasarkan informasi tersebut, bot memutuskan apakah harus mengejar berlian, menghindari rintangan, atau kembali ke base, lalu menentukan arah gerakan menuju tujuan tersebut.

f. Fungsi Objektif (Objective Function)

Seluruh strategi digerakkan oleh tujuan utama yaitu mengumpulkan sebanyak mungkin berlian dalam waktu yang efisien, lalu kembali ke base sebelum waktu habis. Untuk itu, setiap keputusan dinilai berdasarkan efektivitasnya dalam meningkatkan skor dan meminimalkan risiko. Bot juga memperhatikan waktu yang tersisa dan jumlah berlian yang dibawa untuk menentukan kapan harus berhenti mengumpulkan dan mulai pulang, sehingga memaksimalkan hasil permainan.

2. Algoritma Greedy Time-Weighted

Algoritma ini merupakan strategi pengambilan keputusan berbasis *greedy* yang dilengkapi dengan pertimbangan waktu (*time-weighted*), dirancang untuk mengarahkan bot dalam mengumpulkan diamond secara efisien dan kembali ke base dengan aman sebelum waktu habis. Strategi ini berfokus pada pengumpulan diamond sebanyak mungkin, namun tetap memperhitungkan sisa waktu dan jumlah diamond yang telah dikantongi untuk menentukan waktu yang tepat untuk kembali. Bot akan selalu memilih jalur tercepat menuju target, termasuk memanfaatkan portal jika dapat mempercepat perjalanan. Selain itu,

strategi ini juga cermat dalam menghindari risiko, seperti menjauhi diamond merah saat kapasitas hampir penuh, serta mampu menyesuaikan rute jika terdapat rintangan di jalur, guna menjaga keselamatan dan efektivitas langkah.

a. Himpunan Kandidat (Candidate Set)

Himpunan kandidat mencakup seluruh objek atau posisi yang dapat dipilih sebagai tujuan pergerakan bot. Ini termasuk semua diamond yang tersedia di papan permainan, tombol spesial yang dapat memberikan efek menguntungkan seperti menambahkan beberapa diamond sekaligus, dan portal yang memungkinkan bot berpindah lokasi dengan cepat. Himpunan ini dibentuk ulang setiap giliran, menyesuaikan dengan kondisi permainan seperti posisi bot, waktu tersisa, dan status objek-objek di sekitar.

b. Himpunan Solusi (Solution Set)

Himpunan solusi merupakan berbagai kemungkinan rute atau aksi yang dapat dilakukan untuk mencapai kandidat-kandidat tersebut. Setiap solusi mewakili urutan langkah dari posisi bot saat ini menuju target tertentu. Solusi bisa berupa jalur langsung menuju diamond, rute menuju tombol spesial, atau lintasan kompleks yang melibatkan penggunaan portal. Semakin kompleks kondisi di papan, semakin beragam pula solusi yang dipertimbangkan.

c. Fungsi Seleksi (Selection Function)

Fungsi seleksi berperan dalam menentukan solusi terbaik dari sekumpulan alternatif yang tersedia. Penilaian dilakukan menggunakan metode *time-weighted score*, yaitu skor yang memperhitungkan nilai objek (misalnya poin dari diamond) dibagi dengan jarak, lalu disesuaikan dengan waktu tersisa. Semakin mendekati akhir permainan, skor akan lebih mengutamakan jarak yang pendek, sedangkan pada awal permainan nilai poin menjadi pertimbangan utama.

d. Fungsi Kelayakan (Feasibility Function)

Fungsi kelayakan berfungsi menyaring kandidat atau solusi yang tidak cocok untuk diambil pada kondisi tertentu. Misalnya, diamond merah akan dihindari jika bot sudah memiliki empat diamond karena risiko kelebihan kapasitas. Begitu pula, jalur yang terlalu jauh atau tidak realistis dicapai dalam waktu tersisa akan dieliminasi. Dengan demikian, fungsi ini memastikan bahwa hanya opsi yang layak dan relevan yang masuk dalam pertimbangan akhir.

e. Fungsi Solusi (Solution Function)

Fungsi solusi bertugas membangun dan menyesuaikan jalur menuju target dengan memperhitungkan hambatan yang ada. Jika terdapat rintangan seperti portal atau diamond merah yang tidak boleh disentuh, fungsi ini akan membuat target sementara untuk menghindari jalur berbahaya sebelum kembali ke target utama. Fungsi ini memastikan bahwa solusi yang dirancang tidak hanya efisien secara jarak, tetapi juga aman dan dapat ditempuh sesuai kondisi permainan.

f. Fungsi Objektif (Objective Function)

Fungsi objektif menjadi panduan keseluruhan dari strategi, yaitu untuk memaksimalkan jumlah poin yang dikumpulkan dan memastikan bot dapat kembali ke base tepat waktu agar poin tersebut dihitung. Tujuan ini dicapai dengan menyeimbangkan antara mengejar peluang poin di lapangan dan pengambilan keputusan yang hati-hati berdasarkan waktu, posisi, dan kapasitas bot. Segala pemilihan aksi dan evaluasi risiko diarahkan untuk memenuhi fungsi objektif ini secara optimal.

3. Algoritma Greedy Risk Aware

Strategi ini mengutamakan pemilihan target (permata atau tombol spesial) yang memberikan nilai tertinggi dengan jarak terpendek, namun tetap mempertimbangkan faktor risiko seperti jumlah permata yang dibawa, waktu tersisa, dan jarak musuh. Jika risiko dinilai terlalu tinggi, bot akan memutuskan kembali ke base untuk menyimpan permata, baik secara langsung maupun melalui portal.

a. Himpunan Kandidat (Candidate Set)

Himpunan kandidat terdiri dari semua objek yang bisa menjadi target bot dalam permainan, termasuk permata yang tersedia, tombol spesial, dan jalur alternatif melalui portal. Setiap kandidat dianggap sebagai peluang yang potensial untuk menambah skor, dan dikumpulkan secara dinamis dalam setiap langkah permainan berdasarkan kondisi terbaru papan permainan dan status bot.

b. Himpunan Solusi (Solution Set)

Himpunan solusi merupakan kumpulan jalur atau urutan gerakan menuju target yang dipilih dari himpunan kandidat. Solusi ini bisa berupa rute langsung ke permata, rute melalui portal, atau bahkan jalur alternatif untuk menghindari

hambatan seperti portal atau permata merah. Setiap solusi mencerminkan strategi bot dalam mencapai target secara efisien dan aman.

c. Fungsi Seleksi (Selection Function)

Fungsi seleksi bertugas memilih kandidat terbaik berdasarkan skor risiko yang disesuaikan. Skor ini dihitung dengan membagi nilai permata terhadap jarak yang harus ditempuh, lalu dikurangi dengan penalti risiko yang dihitung berdasarkan kedekatan musuh. Dengan pendekatan ini, bot tidak hanya mengejar nilai tertinggi, tetapi juga mempertimbangkan keamanan saat menuju ke target tersebut.

d. Fungsi Kelayakan (Feasibility Function)

Fungsi kelayakan menyaring kandidat yang tidak layak atau terlalu berisiko untuk dikejar. Misalnya, bot akan menghindari permata merah jika sudah membawa 4 permata, karena pengambilan permata merah akan meningkatkan risiko kehilangan semuanya. Selain itu, fungsi ini juga mempertimbangkan rintangan di jalur, dan akan menyesuaikan target jika ada hambatan seperti portal atau permata merah di lintasan.

e. Fungsi Solusi (Solution Function)

Fungsi solusi mengubah target yang telah dipilih menjadi langkah konkret, yaitu arah gerakan dalam bentuk koordinat (delta x dan y). Jika ada hambatan dalam jalur menuju target, bot akan memilih titik antara (intermediate target) terlebih dahulu. Bila tidak ada target yang aman, bot akan melakukan gerakan acak yang relatif aman berdasarkan posisi musuh, untuk tetap aktif namun menghindari bahaya.

f. Fungsi Objektif (Objective Function)

Fungsi objektif dari strategi ini adalah memaksimalkan skor permainan dengan cara mengumpulkan permata bernilai tinggi secara efisien, namun tetap mempertahankan keamanan bot. Hal ini dilakukan dengan menyeimbangkan nilai permata terhadap jarak dan risiko, serta mengatur keputusan kapan harus kembali ke base untuk menyimpan permata berdasarkan ambang risiko yang adaptif terhadap jumlah permata yang dibawa.

4. Algoritma Greedy Value to Distance

Strategi ini mengambil keputusan dengan memilih permata atau tombol spesial terdekat dan bernilai tinggi, sambil memperhitungkan kondisi kritis seperti jumlah permata yang dibawa, sisa waktu, kedekatan base, serta rintangan di jalur seperti portal atau musuh. Jika dianggap berisiko atau tidak efisien untuk melanjutkan pencarian, bot akan memutuskan untuk segera kembali ke base, baik secara langsung atau melalui portal.

a. Himpunan Kandidat (Candidate Set)

Himpunan kandidat dalam strategi ini mencakup semua target potensial yang dapat dikunjungi oleh bot, yaitu permata biasa, tombol spesial (yang bernilai tinggi), serta jalur alternatif melalui portal. Target-target ini dianalisis berdasarkan kombinasi antara nilai yang diberikan dan jarak dari posisi bot. Semua kandidat dikumpulkan secara dinamis sesuai kondisi terkini dari peta permainan..

b. Himpunan Solusi (Solution Set)

Himpunan solusi terdiri dari berbagai rute menuju kandidat terpilih, baik rute langsung maupun rute kompleks melalui portal. Bot mempertimbangkan rute-rute ini sebagai solusi untuk mencapai target dengan risiko minimal dan waktu tempuh yang efisien. Dalam beberapa kasus, solusi juga mencakup jalur alternatif untuk menghindari hambatan seperti portal atau permata merah di tengah jalan.

c. Fungsi Seleksi (Selection Function)

Fungsi seleksi dalam strategi ini bertugas memilih kandidat terbaik berdasarkan nilai per unit jarak—yakni nilai permata dibagi dengan total jarak tempuh yang diperlukan. Dalam kasus rute melalui portal, nilai ini tetap diperhitungkan dengan menambahkan jarak dari posisi bot ke portal dan dari portal ke target. Strategi ini akan memilih kandidat dengan nilai per jarak terendah (semakin kecil rasio, semakin optimal).

d. Fungsi Kelayakan (Feasibility Function)

Fungsi kelayakan berfungsi menyaring kandidat yang tidak aman atau tidak efisien untuk diambil. Contohnya, jika bot telah membawa 4 permata, maka permata merah (bernilai 2) akan dihindari karena akan memicu risiko kehilangan semua. Selain itu, fungsi ini juga memeriksa apakah jalur menuju target melintasi portal atau rintangan lain, dan jika ya, maka akan dipilih titik perantara (intermediate target) untuk menghindarinya.

e. Fungsi Solusi (Solution Function)

Fungsi solusi menerjemahkan target yang telah dipilih menjadi langkah konkret dalam bentuk gerakan pada grid permainan. Dengan menggunakan get_direction, bot menghitung arah terbaik dari posisi saat ini menuju target yang ditentukan. Bila tidak ada target yang aktif atau target gagal dicapai, maka bot akan bergerak secara bergiliran (rotasi arah) untuk tetap menjelajahi area permainan secara pasif.

f. Fungsi Objektif (Objective Function)

Fungsi objektif dalam strategi ini adalah memaksimalkan jumlah skor yang dikumpulkan sambil menghindari risiko kehilangan permata akibat waktu habis, posisi musuh yang terlalu dekat, atau bahaya di jalur. Hal ini dilakukan dengan terus menyesuaikan strategi berdasarkan kondisi permainan: jika membawa banyak permata, waktu hampir habis, atau base berada cukup dekat, maka bot akan memprioritaskan kembali ke base untuk menyimpan permata sebelum melanjutkan pencarian lainnya.

C. Analisis Efisiensi dan Efektivitas Alternatif Solusi

Berdasarkan rancangan algoritma yang telah dibuat, telah dilakukan proses pengetesan untuk menilai seberapa baik rancangan algoritma bekerja. Proses pengetesan akan dilakukan dengan proses- proses sebagai berikut:

1. Bot dijalankan single player pada board sebanyak 5 kali dan dihitung rata-rata diamonds yang didapatkan. Game akan dijalankan selama 60 detik dengan delay tiap gerakannya adalah 1 detik.
2. Keempat bot akan dijalankan bersamaan dan akan dinilai berdasarkan leaderboard yang tersedia di dalam game. Tes ini akah dilakukan sebanyak 5 kali.

Dengan proses diatas, diharapkan efektivitas dari setiap bot dapat dinilai secara objektif. Berikut adalah hasil pengetesan efisiensi yang telah dilakukan terhadap algoritma greedy Cost efficient path, algoritma greedy Time weighted, algoritma greedy Risk aware dan algoritma greedy Value to distance yang dijalankan secara single player serta hasil Ketika keempat bot dijalankan bersamaan.

Tabel 3. 1 Hasil Tes Algoritma Cost Efficient Path

Percobaan	Diamonds yang diperoleh

1	14
2	12
3	14
4	10
5	8
Rata-rata	11,60

Tabel 3. 2 Hasil Tes Algoritma Time Weighted

Percobaan	Diamonds yang diperoleh
1	10
2	18
3	13
4	14
5	10
Rata-rata	13,00

Tabel 3. 3 Hasil Tes Algoritma Risk Aware

Percobaan	Diamonds yang diperoleh
1	9
2	2
3	9
4	9
5	5

Rata-rata	6,80

Tabel 3. 4 Hasil Tes Algoritma Value to Distance

Percobaan	Diamonds yang diperoleh
1	8
2	8
3	8
4	8
5	8
Rata-rata	8,00

Tabel 3. 5 Hasil Tes Perbandingan Seluruh Algoritma Single Player

Percobaan	Diamonds yang diperoleh			
	Cost Effiecint Path	Time Weighted	Simplified Risk Aware	Value to Distance
1	14	10	9	8
2	12	18	2	8
3	14	13	9	8
4	10	14	9	8
5	8	10	5	8
Rata-rata	11,60	13,00	6,80	8,00

D. Strategi Greedy Terpilih

Berdasarkan hasil pengetesan yang telah dilakukan, strategi greedy yang dipilih adalah Greedy Time Weighted. Alasan dipilihnya algoritma ini adalah sebagai berikut:

1. Dalam pengetesan single player yang telah dilakukan, algoritma ini mendapatkan score yang lebih banyak daripada algoritma lain dengan rata rata 13,00.

2. Algoritma ini telah memanfaatkan hampir semua elemen yang ada di dalam game Diamonds. Elemen- elemen tersebut termasuk red button dan teleporter. Red button akan banyak digunakan oleh bot dengan tujuan untuk merugikan lawan dan teleporter digunakan untuk mempersingkat jarak untuk mendapatkan diamonds atau jarak untuk kembali ke base.

3. Algoritma ini mempertimbangkan waktu permainan dalam pengambilan keputusan. Saat waktu tersisa sedikit, bot akan lebih berhati-hati dan cenderung segera kembali ke base jika membawa cukup diamond, sehingga meminimalkan risiko kehilangan poin.

4. Pemilihan diamond oleh algoritma ini tidak hanya berdasarkan jarak atau nilai poin saja, tetapi juga disesuaikan dengan waktu yang ada. Semakin sedikit waktu tersisa, maka semakin diprioritaskan diamond yang lokasinya lebih dekat, agar tidak membuang waktu di jalur yang terlalu jauh.

5. Strategi ini juga menyesuaikan dengan kondisi di permainan. Jika waktu tersisa sudah berada di bawah ambang batas tertentu dan bot belum kembali, algoritma akan memaksa bot segera pulang dengan perhitungan aman, sehingga tetap efisien di akhir permainan.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

A. Implementasi Algoritma Greedy

```
class GreedyDiamondLogic(BaseLogic: BaseLogic)
{ class utama implementasi algoritma Greedy berbasis waktu
  untuk mengambil/mencari diamond dan kembali ke base }

KAMUS LOKAL
static shared_targets: array of Position
static shared_portal_target: GameObject
static shared_intermediate_target: Position
static shared_return_via_portal: boolean

procedure __init__()
function next_move(player_bot: GameObject, board: Board) -> (integer, integer)

function calculate_urgency_threshold(time_ratio: real, diamond_count: integer) ->
integer
function should_return_early(bot_stats, time_ratio: real) -> boolean
function calculate_minimum_return_time() -> integer
function evaluate_base_proximity_time_weighted(time_ratio: real) -> boolean

procedure locate_closest_diamond_time_weighted()
function calculate_time_weighted_score(points: integer, distance: integer,
time_ratio: real) -> real

function find_closest_diamond_direct_time_weighted(time_ratio: real) -> (real,
Position)
function find_closest_diamond_via_portal_time_weighted(time_ratio: real) -> (real,
array of Position, GameObject)
function find_closest_special_button_time_weighted(time_ratio: real) -> (real,
Position)

function is_diamond_collectible(gem: GameObject) -> boolean

function determine_optimal_base_route() -> Position
function calculate_base_distance_via_portal() -> integer
function locate_nearest_portal() -> (Position, Position, GameObject)
function locate_paired_portal(portal: GameObject) -> Position

procedure check_path_obstacles(obstacle_type: string, start_x: integer, start_y:
integer, target_x: integer, target_y: integer)
```

```
{ prosedur initiation untuk class GreedyDiamondLogic }

KAMUS LOKAL
movement_vectors: array of (integer, integer)
target_location: Position
current_heading: integer
calculated_distance: integer

ALGORITMA
movement_vectors <- [(1,0), (0,1), (-1,0), (0,-1)]
target_location <- Nil
current_heading <- 0
calculated_distance <- 0
```

```
function calculate_urgency_threshold(time_ratio, diamond_count)

KAMUS LOKAL
base_threshold: integer

ALGORITMA
base_threshold <- 8000
RETURN base_threshold * (1 + 0.3 * diamond_count) * (1 + (1 - time_ratio) * 0.5)
```

```
function should_return_early(bot_stats, time_ratio)

KAMUS LOKAL
min_return_time: integer
time_left_ms: integer
safety_factor: real

ALGORITMA
IF bot_stats.diamonds == 0 THEN
    RETURN False
min_return_time <- calculate_minimum_return_time()
time_left_ms <- bot_stats.milliseconds_left
safety_factor <- 1.2 + (bot_stats.diamonds * 0.1)
RETURN time_left_ms < (min_return_time * 1000 * safety_factor)
```

B. Penjelasan Struktur Data

1. Position

   Program ini menggunakan beberapa struktur data yang telah tersedia di dalam game dan beberapa variabel yang memanfaatkan struktur data tersebut. Berikut adalah rincian struktur data dan variabel yang dipakai dalam program.

```
class Position:
    x: int
    y: int
```

Kelas ini untuk menyimpan informasi posisi dalam permainan,yang terdiri dari kordinasi x dan y.

2. Game Objek

Representasi dari semua entitas dalam permainan seperti bot, diamond, portal, dan tombol.

```
class GameObject:
    id: str
    type: str
```

```
    position: Position
    properties: Any
```

Properti khusus pada bot (player_bot.properties):

```
class BotProperties:
    base: Position
    diamonds: int
    milliseconds_left: int
```

Properti pada diamond (diamond.properties):

```
class DiamondProperties:
    points: int
```

3. Variabel

Berikut adalah penjelasan dari masing-masing variabel dan fungsinya dalam program:

1. movement_vectors adalah daftar arah gerakan dasar yang tersedia untuk bot, terdiri dari empat tuple yang mewakili arah kanan, bawah, kiri, dan atas. Ini digunakan saat bot tidak memiliki target dan bergerak secara acak. Nilai defaultnya adalah [(1, 0), (0, 1), (-1, 0), (0, -1)].
2. target_location adalah posisi utama yang sedang dituju oleh bot. Ini bisa berupa diamond, tombol spesial, portal, atau base, tergantung pada strategi saat ini. Jika tidak ada tujuan, nilainya adalah None.
3. current_heading adalah indeks dalam movement_vectors yang menunjukkan arah terakhir atau arah saat ini ketika bot sedang bergerak secara acak. Nilai ini akan bertambah setiap kali bot tidak memiliki target, untuk memastikan gerakan berputar.
4. calculated_distance menyimpan jarak ke target terakhir yang dihitung oleh fungsi pencarian diamond. Ini digunakan untuk mengevaluasi apakah jarak ke base sudah cukup dekat dibandingkan dengan target lain, agar bot bisa memutuskan kembali lebih awal.
5. shared_targets adalah daftar posisi yang disimpan secara global di class. Ini digunakan untuk menyimpan urutan target yang sedang ingin dicapai oleh bot. Contohnya, ketika menggunakan portal untuk pulang ke base, shared_targets bisa berisi dua posisi: posisi portal dan posisi base.
6. shared_portal_target adalah objek portal (GameObject) yang sedang menjadi target. Ini digunakan ketika rute terbaik ke base melewati portal. Jika bot tidak sedang merencanakan lewat portal, nilainya akan di-reset menjadi None.
7. shared_intermediate_target adalah titik antara (intermediate) yang harus dicapai terlebih dahulu sebelum melanjutkan ke target_location. Titik ini biasanya ditentukan ketika ada rintangan seperti diamond merah atau portal yang menghalangi jalur langsung ke target.
8. shared_return_via_portal adalah boolean yang menandakan apakah perjalanan menuju base saat ini direncanakan untuk dilakukan melalui portal. Jika nilainya True, maka shared_portal_target dan shared_targets akan digunakan dalam rute pulang.

9. bot_stats adalah shortcut untuk mengambil properti dari player_bot. Di dalamnya terdapat data penting seperti posisi base (bot_stats.base), jumlah diamond yang dibawa (bot_stats.diamonds), dan sisa waktu bermain (bot_stats.milliseconds_left).
10. available_diamonds adalah daftar semua objek diamond yang tersedia di papan permainan. Objek-objek ini nantinya akan dipilih berdasarkan skor dan jaraknya untuk ditentukan sebagai target pengambilan.
11. all_bots adalah daftar seluruh bot yang berada di permainan, termasuk bot pemain dan lawan.
12. opponent_bots adalah daftar bot lawan, yaitu semua bot yang bukan bot pemain (player_bot). Ini digunakan untuk membandingkan kondisi bot kita dengan kondisi lawan.
13. portal_objects adalah daftar semua portal (TeleportGameObject) yang tersedia di permainan. Portal digunakan untuk berpindah tempat secara cepat dan biasanya menjadi bagian dari strategi kembali ke base.
14. special_buttons adalah daftar semua tombol spesial (DiamondButtonGameObject) yang ketika diinjak akan memberikan diamond tambahan. Bot akan mempertimbangkan tombol ini sebagai target potensial saat waktu cukup.
15. opponent_diamonds adalah daftar jumlah diamond yang sedang dibawa oleh semua bot lawan. Ini digunakan untuk analisis kompetitif, meskipun tidak secara eksplisit dipakai dalam logika utama.
16. time_left_ratio adalah rasio antara waktu yang tersisa (milliseconds_left) dengan total waktu permainan (30.000 milidetik). Nilai ini digunakan untuk menyesuaikan strategi: di awal permainan bot cenderung fokus mengambil diamond bernilai tinggi, sementara di akhir permainan akan lebih konservatif.
17. urgency_threshold adalah nilai ambang waktu yang dihitung berdasarkan time_left_ratio dan jumlah diamond yang dibawa. Jika waktu tersisa lebih kecil dari threshold ini, maka bot harus mempertimbangkan untuk kembali ke base.
18. move_x dan move_y adalah dua nilai integer yang menunjukkan arah gerak horizontal dan vertikal yang diputuskan bot untuk satu langkah. Nilai ini dihitung berdasarkan posisi sekarang dan target_location, atau dari movement_vectors saat bergerak acak.

C. Analisis Desain Solusi

Pengujian pada algoritma pertama tama dilakukan dengan melakukan tes keoptimalan bot pada mengambil diamond dalam waktu 60 detik tanpa musuh.Berikut adalah hasil pengetesan yang dilakukan.

Tabel 4. 1 Hasil Pengujian Singlebot Menggunakan Algoritma Time-Weighted

Percobaan	Diamond yang diperoleh
1	10

2	18
3	13
4	14
5	10
Rata rata	13,00

Berdasarkan hasil tersebut,bisa disimpulkan bahwa bot bisa bekerja cukup optimal dalam mengumpulkan diamond deengan rata rata diamond yang di kumpulkan adalah 13. Bot juga bekerja tanpa ada bug. Bot juga akan menggunakan teleporter jika di perlukan. Bot juga dapat memperhitungkan Langkah selanjutnya dengan cepat dengan bergerak satu Langkah setiap satu detik.

Tes selanjutnya akan dilakukan dengan melawan bot lain. Walaupun bot dapat berjalan dengan baik saat di lakukan percobaan sendiri,bukan berarti bot akan berjalan dengan optimal ketika dijalankan pada board dengan bot bot lain. Dalam tes ini akan dilakukan dengan cara memainkan bot bersamaan dengan 3 bot lain. Bot yang digunakan menggunakan logic cost efficient,risk aware,dan value to distance.

Tabel 4. 2 Hasil Pengujian Multibot dalam satu permainan

Percobaan	Diamonds yang diperoleh			
	Time Weighted	Cost Efficient	Risk Aware	ValueToDistance
1	11	0	15	6
2	12	7	8	8
3	9	13	7	12
4	14	5	10	4
5	11	3	10	5
Rata rata	11,4	5,6	10	7

Berdasarkan hasil yang didapat, bot sudah berjalan dengan cukup optimal dan berhasil mengambil diamond paling banyak pada percobaan 2,4, dan 5 hasil percobaan ini cukup bervariasi karena diamond dan base di generate secara acak sehingga ada kemungkinan bot tertentu memiliki keuntungan dan juga kekurangan.

Secara keseluruhan, algoritma bot ini sudah menunjukkan kemampuan yang baik dalam mengumpulkan diamonds dan menghindari obstacle saat bermain tanpa musuh. Namun, dalam game yang sesungguhnya ketika ada kehadiran bot musuh, performa bot menjadi lebih bervariasi. Untuk meningkatkan performa dan kestabilan bot, perlu dilakukan pengembangan lebih lanjut terutama dalam menangani skenario-skenario yang tidak terduga seperti terkena tackle oleh bot musuh.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Tugas besar ini, telah berhasil dikembangkan sebuah bot permainan *Diamonds* dengan memanfaatkan algoritma greedy sebagai strategi utama. Beberapa pendekatan greedy yang diimplementasikan mencakup strategi Cost Efficient Path, Time Weighted, Risk Aware, dan Value to Distance, masing-masing dengan pertimbangan objektif, seleksi kandidat, kelayakan jalur, serta mekanisme evaluasi risiko. Berdasarkan hasil pengujian, strategi *Greedy Time Weighted* menunjukkan performa terbaik dalam mengumpulkan diamond, baik saat dijalankan secara tunggal maupun ketika bersaing dengan bot lain, dengan rata-rata perolehan diamond sebesar 13 per permainan. Hal ini menunjukkan bahwa pendekatan yang memperhitungkan sisa waktu dan efisiensi pergerakan dapat secara signifikan meningkatkan kinerja bot. Selain itu, bot telah mampu memanfaatkan elemen permainan seperti *teleporter* dan *red button*, serta menghindari kondisi berisiko seperti tackle dari musuh. Secara keseluruhan, penerapan algoritma greedy pada game ini menunjukkan efektivitas dan efisiensi dalam pengambilan keputusan berbasis heuristik.

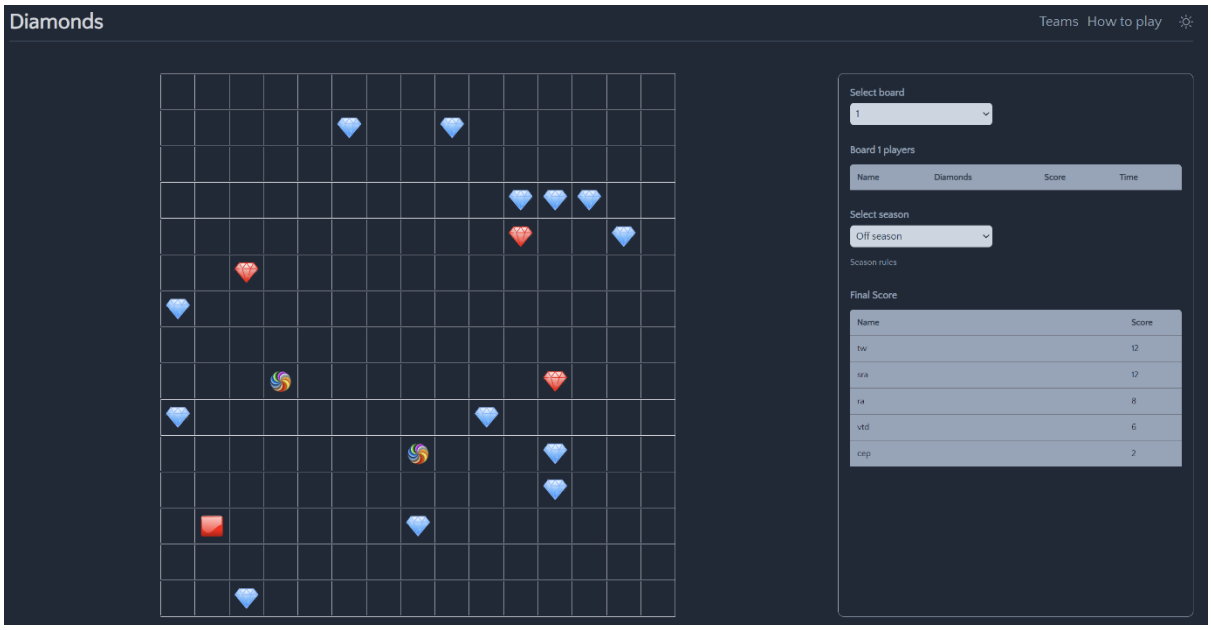
B. Saran

Meskipun bot telah menunjukkan performa yang baik, terdapat beberapa aspek yang dapat ditingkatkan untuk pengembangan ke depan. Salah satu di antaranya adalah kemampuan adaptasi bot terhadap strategi musuh secara real-time, misalnya dengan menyesuaikan prioritas target atau menghindari area padat lawan. Selain itu, diperlukan mekanisme yang lebih responsif dalam menangani situasi darurat seperti inventory penuh saat jauh dari base, atau ketika posisi bot terlalu dekat dengan musuh. Penggabungan algoritma greedy dengan pendekatan lain juga bisa menjadi arah eksplorasi berikutnya untuk meningkatkan performa dalam lingkungan yang lebih dinamis dan kompetitif. Evaluasi tambahan terhadap stabilitas skor di berbagai konfigurasi map dan skenario permainan juga disarankan untuk mendapatkan hasil yang lebih komprehensif.

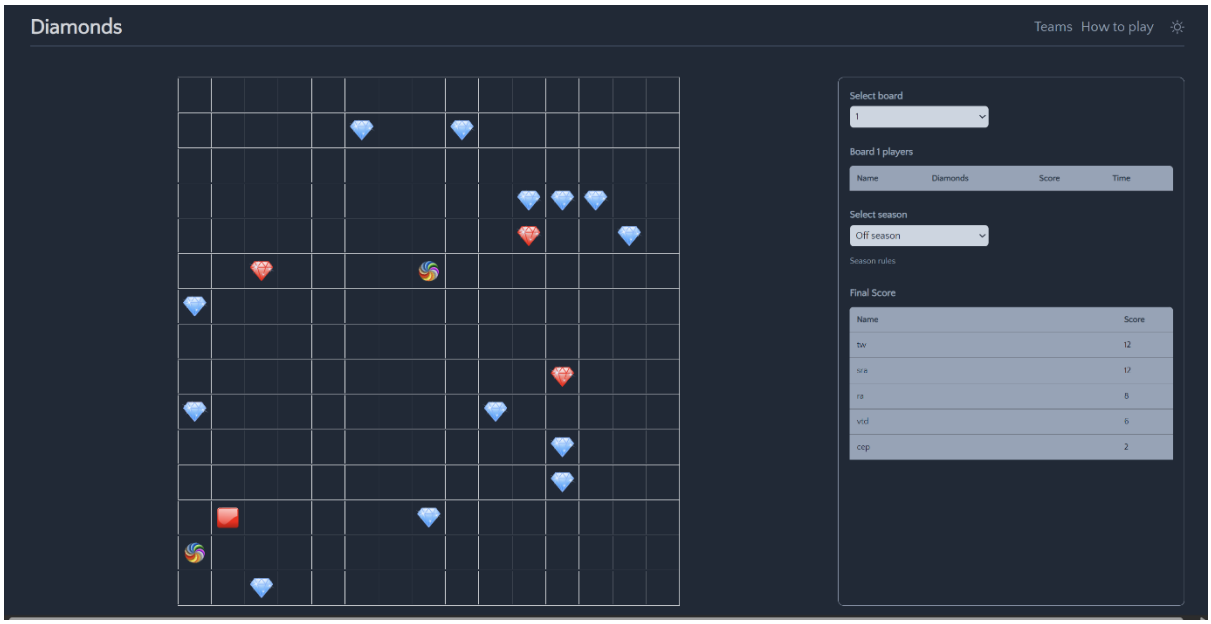
LAMPIRAN

Tautan repository GitHub : https://github.com/Ramaa-Devs/Tubes1_Void

Tautan Video : https://youtu.be/0tyfKFWBmAY

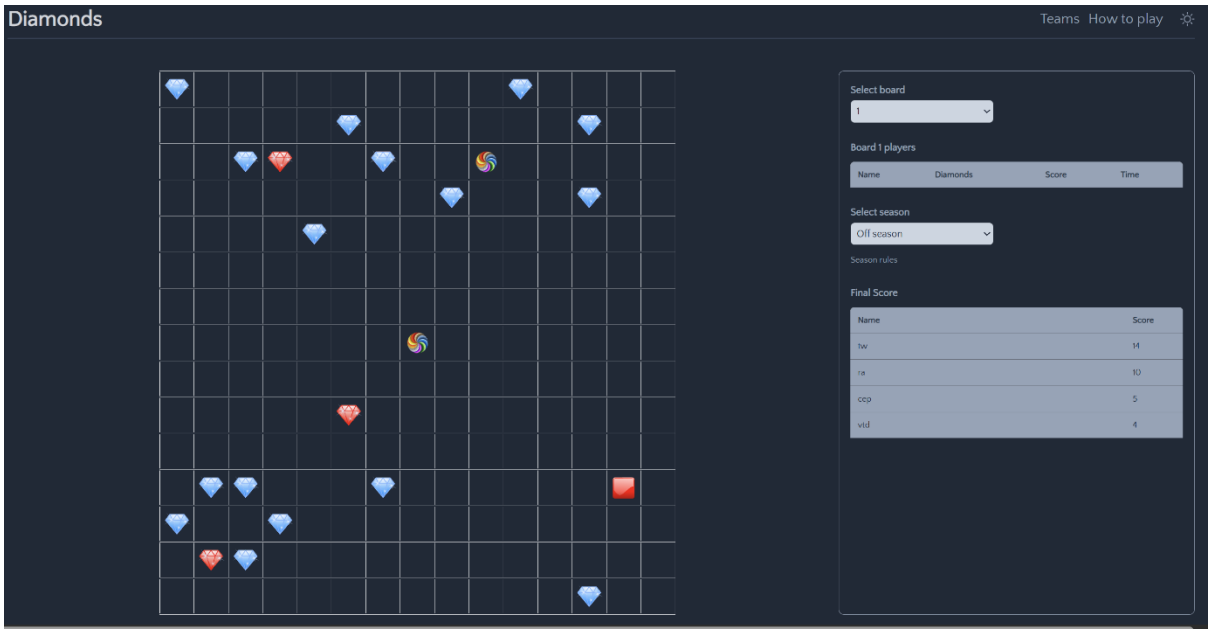


Lampiran 1. Pengujain Multibot percobaan 1

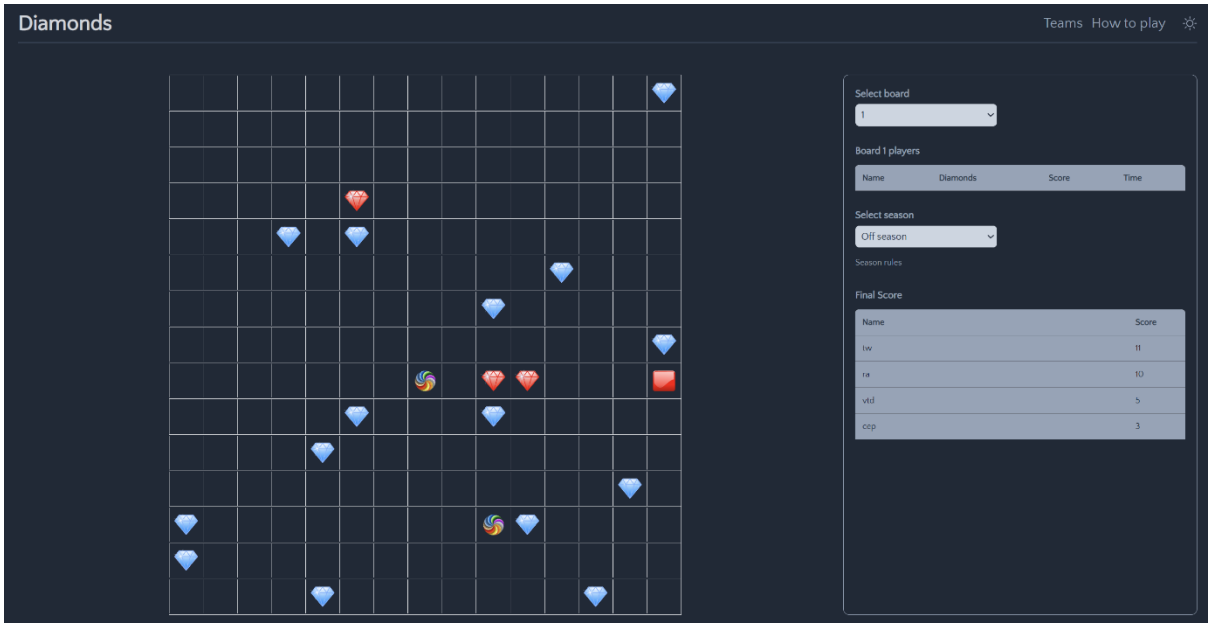


Lampiran 2. Pengujian Multibot Percobaan 2

Lampiran 3. Pengujian Multibot Percobaan 3



Lampiran 4. Pengujian Multibot Percobaan 4



Lampiran 5. Pengujian Multibot Percobaan 5

DAFTAR PUSTAKA

Munir, R. (2021). *Algoritma Greedy (Bagian 1).* Bandung: Sekolah Teknik Elektro dan Informatika ITB.

Munir, R. (2021). *Algoritma Greedy (Bagian 2).* Bandung: Sekolah Teknik Elektro dan Informatika ITB.

Munir, R. (2021). *Algoritma Greedy (Bagian 3).* Bandung: Sekolah Teknik Elektro dan Informatika ITB.