

IMPLEMENTASI ALGORITMA GREEDY DALAM PEMECAHAN BOT PERMAINAN DIAMOND

Tugas Besar



Oleh: Kelompok 9 (CTA BOT)

Pradana Figo Ariasya	123140063
Miftahul Khoiriyah	123140064
Elfa Noviana Sari	123140066

Dosen Pengampu: Winda Yulita, M.Cs.

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

DAFTAR ISI

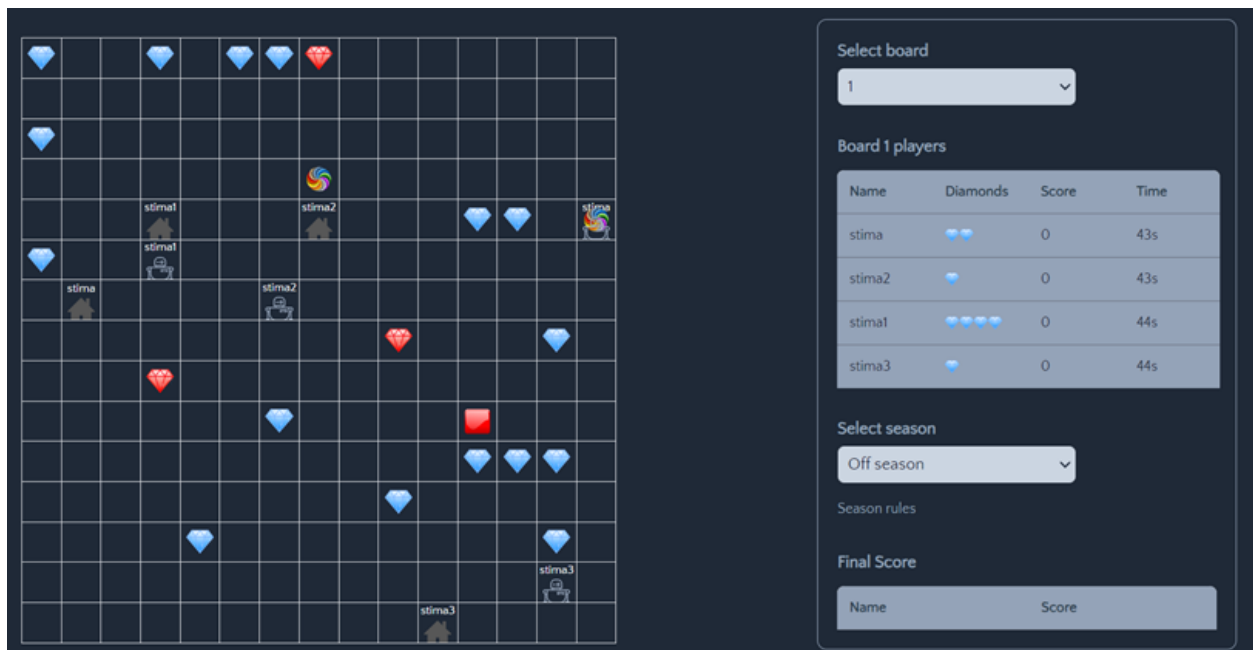
BAB I.....	4
BAB II	11
2.1 Dasar Teori.....	11
2.1.1 Algoritma Greedy	11
2.1.2 Game Engine Etimo Diamonds	13
2.2 Cara Kerja Program	14
2.3 Cara Implementasi Program	15
2.4 Menjalankan Bot Program	16
2.5 Cara mengimplementasikan bot.....	17
BAB III.....	18
3.1 Proses Mapping.....	18
3.1.1 Alternatif Greedy by Shortest Distance Concerning Highest Diamond Value.....	20
3.1.2 Alternatif Greedy by Surrounding Area	21
3.1.3 Alternatif Greedy by Attack Enemy	22
3.1.4 Alternatif Greedy by Closest to Base.....	23
3.1.5 Alternatif Greedy All In Diamond.....	24
3.2 Eksplorasi Alternatif Solusi Greedy	25
3.2.1 Alternatif Greedy by Shortest Distance Concerning Highest Diamond Value	25
3.2.2 Alternatif Greedy by Surrounding Area.....	26
3.2.3 Alternatif Greedy by Attack Enemy	26
3.2.4 Alternatif Greedy by Closest to Base.....	26
3.2.5 Alternatif Greedy All In Diamond.....	27
3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy.....	27
3.3.1 Alternatif Greedy by Shortest Distance Concerning Highest Diamond Value	27
3.3.2 Alternatif Greedy by Surrounding Area.....	28
3.3.3 Alternatif Greedy by Attack Enemy	28
3.3.4 Alternatif Greedy by Closest to Base.....	29
3.3.5 Alternatif Greedy All In Diamond.....	30
3.4 Strategi Greedy yang Dipilih	30
BAB IV	32
4.1 Implementasi Algoritma Greedy.....	32
4.1.1 Kode.....	32
4.1.2 Penjelasan Alur Program	35
4.2 Struktur Data yang Digunakan.....	39
4.3.1 Skenario Pengujian	40
4.3.2 Hasil Pengujian dan Analisis	45

BAB V	48
5.1 Kesimpulan	48
5.2 Saran	48
LAMPIRAN	49
DAFTAR PUSTAKA	50

BAB I

DESKRIPSI TUGAS

Diamonds adalah tantangan pemrograman yang menantang para peserta untuk bertanding menggunakan bot yang mereka kembangkan melawan bot peserta lain. Setiap peserta akan memiliki bot dengan tujuan utama mengumpulkan sebanyak mungkin diamond. Namun, tantangan ini tidak akan mudah karena berbagai hambatan akan menambah keseruan dan kerumitan dalam permainan. Untuk memenangkan kompetisi, peserta harus menerapkan strategi khusus pada bot mereka masing-masing. Penjelasan lebih lanjut mengenai aturan permainan akan dijelaskan di bawah.



Gambar 1.1 Layar permainan Diamonds

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.

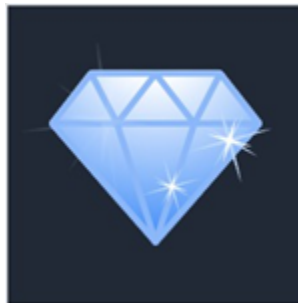
Program permainan Diamonds terdiri atas :

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

Untuk meraih kemenangan dalam pertandingan, pemain harus mengumpulkan sejumlah diamond dengan cara melewati atau mengambilnya. Ada dua varian diamond dalam permainan: diamond merah dan biru. Setiap diamond merah memiliki nilai dua poin, sementara diamond biru bernilai satu poin. Diamond akan terus muncul kembali secara periodik, dan proporsi antara diamond merah dan biru akan berubah-ubah setiap kali terjadi regenerasi.



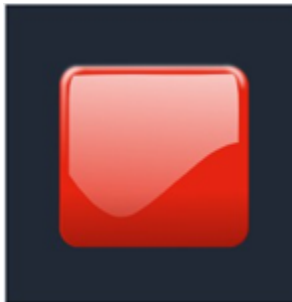
Gambar 1.2 Varian Diamond Biru



Gambar 1.3 Varian Diamond Merah

2. Red Button / Diamond Button

Saat tombol merah dilewati atau dilangkahi, semua diamond, diamond merah maupun diamond biru, akan muncul kembali di papan permainan dengan posisi yang acak. Lokasi tombol merah itu sendiri juga akan berubah ke posisi acak setelah tombol tersebut diaktifkan.



Gambar 1.4 Red Button / Diamond Button

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.



Gambar 1.5 Teleporters

4. Bots

Pada game ini kita akan menggerakkan bot untuk mendapatkan *diamond* sebanyak-banyaknya. Bot ini bisa men-*tickle* bot lainnya.



Gambar 1.6 Bot dengan name “stima”

5. Bases

Setiap bot dilengkapi dengan sebuah Base, yang berfungsi sebagai tempat untuk menampung *diamond* yang dikumpulkan. Ketika *diamond* disimpan di Base, skor dari bot tersebut akan meningkat sesuai dengan nilai *diamond* yang disimpan, dan inventaris bot (yang akan diuraikan lebih lanjut) akan dikosongkan.

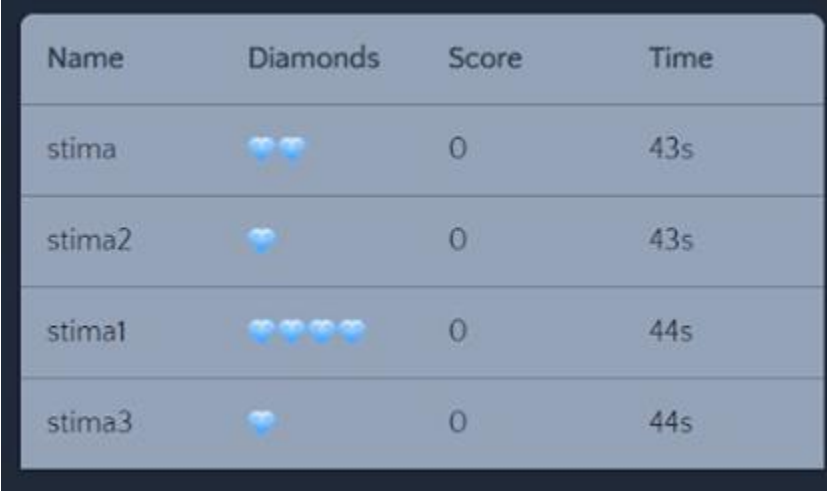


Gambar 1.7 Base dengan name “stima”

6. Inventory

Bot dilengkapi dengan sebuah inventaris, yang berperan sebagai lokasi penyimpanan sementara untuk *diamond* yang telah dikumpulkan. Inventaris ini dibatasi oleh sebuah kapasitas maksimal, yang artinya dapat terisi penuh. Untuk menghindari

keadaan penuh tersebut, bot dapat mentransfer isi inventaris mereka ke Base, sehingga memungkinkan inventaris tersebut untuk dikosongkan kembali.



Name	Diamonds	Score	Time
stima	2	0	43s
stima2	1	0	43s
stima1	5	0	44s
stima3	1	0	44s

Gambar 1.8 Layar Inventory

Untuk mengetahui *flow* dari game ini, berikut 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 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.

Dalam tugas besar ini, mahasiswa diminta untuk bekerja dalam kelompok minimal dua orang dan maksimal tiga orang, dengan lintas kelas dan lintas kampus diperbolehkan. Tujuan dari tugas ini adalah untuk mengimplementasikan algoritma Greedy pada bot permainan Diamonds dengan strategi yang dirancang untuk memenangkan permainan. Setiap kelompok diminta untuk mengembangkan strategi Greedy terbaik yang berkaitan dengan fungsi objektif permainan, yaitu mengumpulkan diamond sebanyak mungkin dan mencegah diamond tersebut diambil oleh bot lain.

Strategi Greedy yang diterapkan oleh setiap kelompok harus dijelaskan secara eksplisit dalam laporan, dan kode program yang sesuai dengan strategi tersebut harus disertakan. Mahasiswa dilarang menggunakan kode program yang diunduh dari internet; setiap kelompok diharapkan untuk membuat program mereka sendiri dengan menggunakan

keaktivitas mereka. Program harus dapat dijalankan pada game engine yang ditentukan dan dapat bersaing dengan bot dari kelompok lain.

Setiap implementasi strategi Greedy harus dijelaskan dengan komentar yang jelas dalam kode program. Selain itu, terdapat bonus poin untuk kelompok yang membuat video tentang aplikasi Greedy pada bot dan simulasinya dalam permainan, dengan menampilkan wajah dari setiap anggota kelompok. Asistensi tugas besar bersifat opsional, namun jika diperlukan, mahasiswa dapat meminta bimbingan dari asisten yang dipilih.

Seluruh laporan tugas besar harus dikumpulkan dalam format PDF melalui tautan yang telah disediakan paling lambat pada tanggal 1 Mei 2025. Isi dari repository yang digunakan untuk menyimpan program harus mengikuti struktur yang telah ditentukan, termasuk folder `src` untuk source code, folder `doc` untuk laporan, dan `README` yang berisi penjelasan singkat algoritma Greedy yang diimplementasikan, persyaratan program, langkah-langkah untuk penggunaan, dan informasi tentang pembuat program.

BAB II

LANDASAN TEORI

2.1 Dasar Teori

2.1.1 Algoritma Greedy

Algoritma greedy merupakan sebuah pendekatan dalam pemrograman dinamis yang memilih solusi terbaik pada setiap langkahnya tanpa mempertimbangkan konsekuensi di masa depan. Pendekatan ini sering digunakan untuk menyelesaikan berbagai masalah optimasi, seperti masalah penjadwalan, penentuan rute, dan lain-lain.

Dalam algoritma greedy, pilihan yang tampaknya terbaik pada saat itu dibuat dengan harapan bahwa pilihan ini akan mengarah pada solusi global yang optimal. Setelah membuat pilihan, algoritma greedy tidak pernah mempertimbangkan pilihan tersebut kembali, artinya algoritma ini tidak memiliki mekanisme backtracking.

Untuk menentukan pilihan yang optimal pada setiap langkah, algoritma greedy menggunakan kriteria tertentu, yang sering kali berupa fungsi untuk memaksimalkan atau meminimalkan nilai tertentu. Masalah yang dapat diselesaikan dengan algoritma greedy sering memiliki substruktur optimal, yang berarti solusi optimal untuk masalah tersebut dapat dibangun dari solusi optimal untuk sub masalahnya.

Sifat pilihan serakah menyatakan bahwa pilihan lokal yang dibuat oleh algoritma greedy dapat diintegrasikan ke dalam solusi global tanpa perlu mempertimbangkan kembali pilihan sebelumnya.

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 :

- a. Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap langkah.
- b. Himpunan solusi, S : berisi kandidat yang sudah dipilih
- c. Fungsi solusi : menentukan apakah himpunan yang dipilih sudah memberikan solusi

- d. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
- e. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih layak atau tidak untuk dimasukkan ke dalam himpunan solusi.
- f. Fungsi objektif: untuk memaksimumkan atau meminimumkan

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 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, pada sebagian persoalan, algoritma greedy tidak selalu berhasil memberi solusi terbaik. Jika solusi terbaik mutlak tidak terlalu diperlukan, maka algoritma greedy dapat digunakan untuk menghasilkan solusi hampiran (approximation), daripada menggunakan algoritma yang kebutuhan waktunya eksponensial untuk menghasilkan solusi yang terbaik. Contohnya adalah permasalahan mencari lintasan dengan bobot minimal pada persoalan Traveling Salesman Problem pada grafik dengan jumlah simpul n . Apabila jumlah simpul sangat banyak, maka penyelesaian dengan algoritma brute force akan membutuhkan waktu komputasi yang lama untuk menemukannya. Namun dengan algoritma greedy, meskipun tur dengan berbobot minimal tidak dapat ditemukan, namun solusi dengan algoritma greedy dianggap sebagai hampiran solusi optimal. Secara matematis, pembuktian bahwa algoritma greedy tidak selalu menghasilkan solusi

yang optimal biasanya dilakukan dengan menggunakan counterexample bahwa ada solusi yang lebih 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 sistem
4. Persoalan knapsack (knapsack problem)
5. Penjadwalan job dengan tenggat waktu (job scheduling with deadlines)
6. Pohon merentang minimum (minimum spanning tree)
7. Lintasan terpendek (shortest path)
8. Kode Huffman (Huffman code)
9. Pecahan Mesir (Egyptian fraction)

2.1.2 Game Engine Etime Diamonds

Game Engine adalah sistem perangkat lunak yang dimanfaatkan untuk mengembangkan atau menciptakan sebuah game. Game engine merupakan *library* yang digunakan untuk membuat game. Game engine etimo diamonds merupakan game yang berasal dari permainan dahulu yaitu catur Go/Baduk. Pada permainan tersebut kita diminta untuk mengepung batu lawan sehingga kita dapat mengambil batu lawan yang dikepung. Tentu hal ini membuat kita ingin mengambil atau mengepung lawan sebanyak-banyaknya dan hal ini pula yang menjadi dasar dari pembuatan Game engine etimo diamonds, yaitu ingin mengambil atau mendapatkan diamonds sebanyak-banyaknya. Game engine etimo diamonds diciptakan untuk menambah tingkat kesulitan permainan dengan menambahkan beberapa fitur dan aturan yang membuat permainan menjadi lebih menarik.

Game engine etimo diamonds tetap mempertahankan latar catur pada permainan Baduk namun yang berubah adalah pada catur permainannya, jika pada permainan Baduk catur yang digunakan hanya berwarna hitam dan putih yang berjumlah 181 batu hitam dan 180 batu putih dan permainan ini hanya terbatas pada dua pemain saja namun Game engine etimo diamonds dapat dimainkan oleh beberapa orang. Game ini sedikit berbeda dengan catur Baduk, jika permainan Baduk akan memenangkan banyak batu lawan, game ini akan dimenangkan berdasarkan banyaknya diamonds

yang didapat. Bukan hanya itu game ini juga memiliki batasan waktu dan permainan tersebut mengandalkan bot yang telah dirancang dengan algoritma Greedy untuk memenangkan permainan. Dapat dikatakan bahwa game ini merupakan pertarungan antar bot.

Game engine etimo diamonds merupakan pertarungan antar bot, dengan batasan waktu, serta beberapa aturan baru. Aturan tersebut adalah jika berhasil mendapatkan Diamonds merah bernilai 2 poin, sedangkan yang biru bernilai 1 poin. Terdapat juga *red button*, ketika *red button* dilewati maka semua diamond akan di-generate kembali pada board dengan posisi acak. Posisi *red button* ini juga akan berubah secara acak jika red button ini dilangkahi. Terdapat juga 2 teleporter yang terhubung satu sama lain, teleport tersebut berbentuk cakram warna-warni, jika teleporter tersebut dilalui maka bot akan berpindah. Hal yang perlu diperhatikan pada game ini adalah sebuah base. Base ini berfungsi untuk menyimpan diamonds yang telah kita dapatkan, apabila diamonds disimpan ke base maka score akan bertambah sebanyak diamonds yang dibawa. 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

2.2 Cara Kerja Program

Program game Diamonds beroperasi dengan urutan langkah-langkah berikut:

1. Inisialisasi Game: Engine permainan diaktifkan dan menjalankan pertandingan di alamat yang ditentukan, seperti localhost:8082 untuk lingkungan lokal.
2. Koneksi Bot: Engine menunggu sampai semua bot pemain terhubung sebelum memulai pertandingan. Proses logging dilakukan untuk merekam kejadian penting selama permainan.
3. Pengaturan Jumlah Bot: Jumlah bot yang berpartisipasi dalam satu pertandingan ditentukan oleh konfigurasi dalam file "appsettings.json".
4. Mulai Pertandingan: Pertandingan dimulai setelah semua bot terhubung dan jumlahnya sesuai dengan pengaturan yang ditetapkan.
5. Interaksi Bot dan Engine: Bot pemain berinteraksi dengan engine, menerima event dari engine dan mengirimkan respons berupa aksi yang akan dilakukan oleh bot.
6. Pelaksanaan Game: Game berlangsung sampai kondisi berakhirnya pertandingan terpenuhi, seperti batas waktu atau kriteria kemenangan. Hasil pertandingan dicatat dalam file JSON untuk analisis lebih lanjut.

7. Note: Program ini mengelola interaksi antara engine dan bot pemain, memfasilitasi pelaksanaan pertandingan Diamonds, dan mendokumentasikan hasilnya.

2.3 Cara Implementasi Program

Cara mengimplementasikan program dengan menjalankan game engine :

a. *Requirement* yang harus di-install

- Node.js

[Node.js](https://nodejs.org/en) dapat diinstal pada link berikut (<https://nodejs.org/en>), berfungsi untuk menghubungkan terminal ke website

- Docker desktop

Docker desktop berfungsi untuk setup local database yang dapat diunduh pada (<https://www.docker.com/products/docker-desktop/>)

- Yarn

Yarn berfungsi sebagai penghubung, yarn dapat diinstal dengan

```
Npm install --global yarn
```

b. Instalasi dan konfigurasi awal

1. Download source code (.zip) pada ([release game engine](#))
2. Extract zip tersebut, lalu masuk ke folder hasil extractnya dan buka terminal
3. Masuk ke root directory dari project (sesuaikan dengan nama rilis terbaru)

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

4. Install dependencies menggunakan Yarn

```
yarn
```

5. Setup default environment variable dengan menjalankan script berikut untuk Windows

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

6. Setup local database (buka aplikasi docker desktop terlebih dahulu, lalu jalankan command berikut di terminal)

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

Lalu jalankan script berikut untuk Windows

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

- c. Jalankan perintah berikut untuk melakukan build frontend dari game-engine

```
npm run build
```

- d. Jalankan perintah berikut untuk memulai game-engine

```
npm run start
```

Notes : Frontend dapat dikunjungi melalui (<http://localhost:8082/>) Perlu diingat bahwa untuk menjalankan game kedua kalinya, hanya perlu membangun frontend dan menyalakan game engine sesuai langkah no c dan d diatas.

2.4 Menjalankan Bot Program

- a. *Requirement* yang harus di-install

- Python (<https://www.python.org/downloads/>)

- b. Instalasi dan konfigurasi awal

1. Download source code (.zip) pada ([release bot starter pack](#))
2. Extract zip tersebut, lalu masuk ke folder hasil extractnya dan buka terminal
3. Masuk ke root directory dari project (sesuaikan dengan nama rilis terbaru)

```
cd tubes1-IF2110-bot-starter-pack-1.0.1
```

4. Install dependencies menggunakan pip

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

c. Jalankan bot

Untuk menjalankan satu bot (pada contoh ini, kita menjalankan satu bot dengan logic yang terdapat pada file game/logic/random.py)

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

Untuk menjalankan beberapa bot sekaligus (pada contoh ini, kita menjalankan 4 bot dengan logic yang sama, yaitu game/logic/[random.py](#)) untuk windows

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

Notes :

1. Jika kalian menjalankan beberapa bot, pastikan setiap email dan nama unik
2. Email bisa apa saja asalkan mengikuti sintaks email yang benar, tidak harus email yang terdaftar (misal stima22@email.com)
3. Nama dan password bisa apa saja tanpa spasi

2.5 Cara mengimplementasikan bot

- a. Buatlah folder baru pada direktori /game/logic (misalnya mybot.py)
- b. Buatlah kelas yang meng-inherit kelas BaseLogic, lalu implementasikan constructor dan method next_move pada kelas tersebut
- c. Import kelas yang telah dibuat pada main.py dan daftarkan pada dictionary CONTROLLERS
- d. Jalankan program seperti step c pada bagian 2 (sesuaikan argumen logic pada command/script tersebut menjadi nama bot yang telah terdaftar pada CONTROLLERS). Anda bisa menjalankan satu bot saja atau beberapa bot menggunakan .bat atau .sh script


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

BAB III

APLIKASI STRATEGI *GREEDY*

3.1 Proses Mapping

Dalam melakukan penerapan algoritma greedy dalam game diamond tersebut, perlu melakukan identifikasi dan pemetaan (mapping) elemen-elemen dalam game ke dalam struktur dasar greedy. Hal ini penting agar strategi yang digunakan tetap mengikuti kerangka kerja algoritma greedy dan mampu menghasilkan solusi yang efisien. Berikut ini adalah elemen-elemen dalam game Diamonds yang dimapping menjadi komponen greedy:

1. Himpunan kandidat (**C**)

Himpunan kandidat dalam permasalahan ini akan mencakup koordinat-koordinat dari berbagai objek yang ada di dalam permainan. Objek-objek tersebut meliputi blue diamond, red diamond, red button, teleporter, base bot, bot musuh, dan base musuh. Selain itu, elemen penting yang perlu diperhatikan dalam game adalah jarak antara bot dan koordinat tujuan. Jarak ini akan mempengaruhi keputusan bot dalam memilih tujuan terbaik, karena semakin dekat tujuan dengan bot, semakin efisien pergerakan bot dalam mengumpulkan berlian atau mencapai target yang diinginkan.

2. Himpunan solusi (**S**)

Himpunan solusi berisi koordinat-koordinat yang memberikan keuntungan maksimal jika didatangi oleh bot. Berdasarkan pendekatan greedy, himpunan solusi ini bisa terdiri dari satu koordinat tunggal yang memberikan keuntungan terbesar, atau beberapa koordinat yang harus dilalui secara berurutan untuk mencapai keuntungan optimal. Proses ini mempertimbangkan faktor-faktor seperti jarak dan nilai keuntungan dari setiap koordinat, sehingga bot akan memilih jalur yang memberikan keuntungan maksimal secara keseluruhan.

3. Fungsi Solusi

Fungsi solusi digunakan untuk menentukan apakah kandidat yang dipilih dapat memberikan solusi yang valid. Secara umum, fungsi ini hanya diterapkan ketika permainan hampir berakhir. Fungsi ini akan menghitung apakah waktu yang tersisa cukup untuk mencapai suatu koordinat di peta dan kembali ke base. Fungsi ini hanya digunakan di akhir permainan karena fungsi-fungsi sebelumnya, seperti fungsi seleksi, sudah memastikan bahwa hanya kandidat yang memenuhi syarat yang dipilih, sehingga kandidat yang tidak dapat menjadi solusi telah dihapus sejak awal.

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 berfungsi untuk memastikan apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi yang valid. Dalam konteks permainan Diamonds, fungsi kelayakan digunakan untuk menentukan apakah objek dalam permainan, seperti diamond, dapat dijadikan tujuan berikutnya bagi bot. Sebagai contoh, jika red diamond dipilih sebagai tujuan selanjutnya, tetapi bot sudah memiliki 4 diamonds dalam inventory-nya, maka red diamond tidak bisa diambil karena bot tidak memiliki cukup ruang untuk menyimpannya. Fungsi kelayakan akan mencegah bot untuk menuju koordinat red diamond tersebut. Jika tidak dicegah, bot bisa terjebak, berputar-putar di sekitar red diamond, atau bahkan menghasilkan gerakan yang tidak valid, yang dapat menyebabkan bug dalam permainan.

6. Fungsi Objektif

Fungsi objektif digunakan untuk memilih kandidat terbaik dari himpunan kandidat dengan tujuan memaksimalkan atau meminimalkan nilai tertentu. Dalam permasalahan ini, fungsi objektif akan diterapkan untuk mengevaluasi dan memilih kandidat yang memberikan keuntungan terbesar. Dengan kata lain, fungsi objektif akan memaksimalkan

keuntungan yang diperoleh dari himpunan kandidat dengan mempertimbangkan faktor-faktor seperti nilai, jarak, atau elemen lain yang relevan untuk menentukan solusi terbaik.

Berikut merupakan penerapan konsep pemetaan elemen-elemen utama dari algoritma greedy pada masing-masing alternatif strategi yang telah kami rancang, yang mencakup identifikasi himpunan kandidat, himpunan solusi, serta penerapan fungsi seleksi, kelayakan, solusi, dan objektif secara sistematis untuk menyesuaikan strategi dengan kondisi dinamis permainan serta memaksimalkan perolehan poin diamond secara efisien :

3.1.1 Alternatif Greedy by Shortest Distance Concerning Highest Diamond Value

Elemen Algoritma Greedy	Pemetaan pada Permainan Diamonds
Himpunan Kandidat (C)	Himpunan kandidat dalam strategi ini akan berisi kumpulan koordinat semua diamonds yang ada pada papan permainan, termasuk posisi bot musuh lain.
Himpunan Solusi (S)	Himpunan solusi berisi koordinat diamond yang memberi keuntungan terbesar saat diambil oleh bot, dengan mempertimbangkan nilai dan jaraknya.
Fungsi Solusi	Fungsi solusi pada strategi ini adalah memastikan diamond yang dipilih sesuai dengan kapasitas inventory, diamond bernilai 2 tidak dipilih jika sisa slot hanya 1.
Fungsi Seleksi (<i>Selection</i>)	Fungsi seleksi dilakukan dengan menghitung rasio antara nilai diamond dengan jaraknya ke bot, dan memilih diamond dengan rasio tertinggi sebagai target utama.
Fungsi Kelayakan (<i>Feasible</i>)	Fungsi kelayakan memastikan hanya diamond yang sesuai kapasitas inventory yang bisa dipilih agar tidak sia-sia saat diambil.
Fungsi Objektif	Fungsi objektif strategi ini memaksimalkan total keuntungan

	dengan mempertimbangkan nilai, jarak, dan memanfaatkan teleporter untuk efisiensi jalur.
--	--

3.1.2 Alternatif Greedy by Surrounding Area

Elemen Algoritma Greedy	Pemetaan pada Permainan Diamonds
Himpunan Kandidat (C)	Himpunan kandidat terdiri dari blok-blok berukuran 3x3 dan 5x5, yang nantinya akan dievaluasi berdasarkan jumlah nilai diamond di dalamnya. Terdapat sembilan blok untuk masing-masing ukuran tersebut.
Himpunan Solusi (S)	Himpunan solusi berisi kumpulan koordinat diamond dalam satu blok yang memberikan keuntungan terbesar saat dikunjungi bot.
Fungsi Solusi	Fungsi solusi menentukan lokasi terbaik bagi bot berdasarkan diamond di blok kandidat, dimulai dari blok 3x3, jika kosong lanjut ke blok 5x5. Diasumsikan selalu ada diamond di salah satu blok tersebut.
Fungsi Seleksi (<i>Selection</i>)	Fungsi seleksi dilakukan dalam dua tahap yaitu pencarian blok 3x3 terlebih dahulu, kemudian blok 5x5. Setiap blok dinilai dengan menjumlahkan nilai diamond di dalamnya dan membaginya dengan jarak diamond ke bot, sehingga didapatkan skor keuntungan.
Fungsi Kelayakan (<i>Feasible</i>)	Fungsi kelayakan memeriksa apakah diamond target bisa diambil sesuai kapasitas inventory bot. Jika diamond merah bernilai dua tapi hanya tersisa satu slot, target dianggap tidak layak dan ditolak.

Fungsi Objektif	Fungsi objektif berperan untuk memaksimalkan keuntungan total dari koordinat solusi dengan mempertimbangkan nilai diamond sekaligus jarak tempuh bot, sehingga solusi yang dipilih adalah yang paling optimal.
-----------------	--

3.1.3 Alternatif Greedy by Attack Enemy

Elemen Algoritma Greedy	Pemetaan pada Permainan Diamonds
Himpunan Kandidat (C)	Himpunan kandidat berisi koordinat base musuh di papan permainan. Base dipilih karena bot bisa menunggu musuh kembali tanpa harus mengejar, mengingat posisi musuh sulit diprediksi.
Himpunan Solusi (S)	Himpunan solusi ini memperkecil ruang keputusan dengan hanya menyertakan musuh yang bisa langsung ditangani. Berisi base musuh yang paling berpeluang untuk ditackle dengan potensi keuntungan tinggi.
Fungsi Solusi	Fungsi solusi dalam strategi ini menjalankan aksi tackle untuk merebut diamond dari bot lawan. Ini menjadi inti strategi serangan, karena ini tindakan paling menguntungkan secara langsung.
Fungsi Seleksi (<i>Selection</i>)	Fungsi seleksi memilih prioritas diberikan pada musuh yang membawa lebih banyak diamond. Kriteria tambahan seperti arah gerak musuh, potensi risiko, dan jarak ke base juga bisa dipertimbangkan.
Fungsi Kelayakan (<i>Feasible</i>)	Fungsi kelayakan dalam strategi ini digunakan untuk menentukan apakah base musuh layak menjadi target. Base dianggap layak jika jarak bot ke base lebih dekat

	dibandingkan jarak musuh pemilik base ke base tersebut
Fungsi Objektif	Fungsi objektif strategi ini bertujuan meminimalkan jarak tempuh bot ke seluruh base musuh dengan menghitung titik tengah dari semua koordinat base.

3.1.4 Alternatif Greedy by Closest to Base

Elemen Algoritma Greedy	Pemetaan pada Permainan Diamonds
Himpunan Kandidat (C)	Himpunan kandidat berisi semua koordinat diamond yang ada di papan permainan. Diamond-diamond ini akan dievaluasi berdasarkan jaraknya terhadap posisi base bot.
Himpunan Solusi (S)	Himpunan solusi ini berisi koordinat satu atau beberapa diamond yang jaraknya paling dekat ke base, namun tetap berada dalam jangkauan bot untuk diambil.
Fungsi Solusi	Fungsi solusi dalam strategi ini memastikan bahwa diamond yang dipilih memiliki nilai yang sesuai dengan kapasitas inventory basenya sendiri.
Fungsi Seleksi (<i>Selection</i>)	Fungsi seleksi mengurutkan seluruh diamond berdasarkan jaraknya ke base bot dan memilih diamond dengan jarak terdekat. Jika ada beberapa diamond dengan jarak sama, bisa diprioritaskan yang lebih dekat ke bot.
Fungsi Kelayakan (<i>Feasible</i>)	Fungsi kelayakan ketika diamond hanya layak dipilih jika dapat dijangkau oleh bot sebelum inventory penuh, serta nilai diamond sesuai dengan sisa slot inventory.
Fungsi Objektif	Fungsi objektif strategi ini bertujuan memaksimalkan efisiensi rute pengumpulan dengan meminimalkan total jarak

	dari diamond ke base
--	----------------------

3.1.5 Alternatif Greedy All In Diamond

Elemen Algoritma Greedy	Pemetaan pada Permainan Diamonds
Himpunan Kandidat (C)	Himpunan kandidat berisi posisi diamond dalam blok 3x3 yang dievaluasi berdasarkan total nilai dan jarak ke bot. Diamond merah diabaikan jika bot sudah membawa 4 diamond. Kandidat akhir adalah posisi terdekat yang layak dijadikan tujuan.
Himpunan Solusi (S)	Solusi greedy ini adalah urutan langkah-langkah gerakan yang akan diambil bot untuk menuju tujuan terpilih dari himpunan kandidat. Langkah-langkah ini disusun secara berurutan berdasarkan arah tercepat menuju tujuan.
Fungsi Solusi	Himpunan solusi algoritma greedy ini berupa jalur perjalanan bot menuju target yang paling menguntungkan. Pemilihan jalur memperhatikan jarak terdekat, manfaat dari target yang dituju, serta menghindari hambatan yang mungkin mengganggu perjalanan bot.
Fungsi Seleksi (<i>Selection</i>)	Fungsi seleksi dalam algoritma greedy ini memilih kandidat dengan nilai tertinggi dan jarak terdekat. Jika sudah penuh, bot langsung kembali ke base begitupula rintangan juga diperhitungkan saat memilih tujuan.
Fungsi Kelayakan (<i>Feasible</i>)	Fungsi kelayakan dalam algoritma greedy ini dianggap layak jika bisa dicapai tanpa hambatan berbahaya. Jika ada rintangan, bot mencari jalur aman yang menyimpang sedikit.

Fungsi Objektif	Fungsi objektif dari algoritma ini adalah mengumpulkan diamond bernilai tinggi seefisien mungkin dan kembali ke base saat penuh, sambil memanfaatkan fitur seperti tombol dan teleporter untuk keuntungan.
-----------------	--

3.2 Eksplorasi Alternatif Solusi Greedy

Dalam permainan diamonds terdapat beberapa bot yang akan dipertandingkan dalam satu match. Bot-bot ini nantinya akan saling bertarung satu sama lain dengan memperebutkan skor tertinggi di setiap match tersebut, dimana skor tertinggi yang akan dicapai melalui banyaknya diamond yang diperoleh oleh setiap algoritma yang ada. Dengan itu kami mengimplementasikan berbagai strategi greedy yang optimal untuk dipilih :

3.2.1 Alternatif Greedy by Shortest Distance Concerning Highest Diamond Value

Algoritma greedy by Shortest Distance Concerning Highest Diamond Value dalam permainan Diamonds menggunakan berbagai alternatif strategi pengambilan keputusan yang tetap berlandaskan prinsip memilih solusi lokal terbaik di setiap langkah. Beberapa alternatif pendekatan yang dapat diterapkan antara lain: memilih diamond terdekat untuk meminimalkan jarak tempuh, memilih diamond dengan nilai tertinggi tanpa mempertimbangkan jarak, memilih berdasarkan rasio nilai terhadap jarak, hingga memilih blok 3x3 atau 5x5 dengan total nilai diamond tertinggi. Selain itu, strategi juga bisa mencakup tindakan menyerang bot lawan yang membawa banyak diamond apabila mereka berada dalam jangkauan. Setiap alternatif tersebut memiliki keunggulan dalam situasi yang berbeda, dan eksplorasi berbagai pendekatan ini memungkinkan pemilihan strategi greedy yang paling efisien tergantung pada kondisi permainan saat itu.

3.2.2 Alternatif Greedy by Surrounding Area

Algoritma greedy by Surrounding Area menggunakan alternatif solusi dengan membagi papan permainan menjadi blok-blok berukuran 3x3 dan 5x5, lalu memilih blok dengan rasio keuntungan tertinggi berdasarkan nilai diamond terhadap jarak ke bot. Alternatif yang dieksplorasi mencakup pemilihan blok terdekat saat nilai blok seimbang, prioritas pencarian lokal di blok kecil sebelum meluas ke blok besar, pengambilan beberapa

diamond dalam satu blok sebagai target terpadu, serta pemanfaatan teleporter untuk mengakses blok bernilai tinggi dengan jarak tempuh yang lebih pendek. Strategi ini juga menyaring diamond yang tidak sesuai dengan kapasitas inventory bot agar tidak diproses lebih lanjut. Pendekatan ini membuat bot dapat secara efisien mengumpulkan diamond dalam jumlah besar tanpa harus berpindah-pindah ke lokasi terpencar.

3.2.3 Alternatif Greedy by Attack Enemy

Alternatif Greedy by Attack Enemy menggunakan alternatif solusi dengan memanfaatkan fitur tackle untuk memperoleh diamond dari bot musuh secara langsung, alih-alih mencarinya sendiri. Pendekatan ini menargetkan base musuh sebagai titik strategis untuk melakukan serangan, dengan seleksi dilakukan terhadap musuh yang membawa diamond terbanyak. Alternatif yang dieksplorasi termasuk memilih base musuh yang paling dekat dan paling menguntungkan, menunggu di base untuk efisiensi gerak, serta menghitung posisi tengah dari semua base musuh untuk menentukan titik pengintaian optimal sejak awal permainan. Strategi ini tidak hanya memaksimalkan perolehan diamond dengan satu langkah, tapi juga mengganggu pergerakan musuh dengan memaksa mereka kembali ke base, sehingga membuka celah penguasaan area dan keuntungan jangka panjang.

3.2.4 Alternatif Greedy by Closest to Base

Strategi Greedy by Closest to Base menggunakan alternatif solusi dengan memprioritaskan pengambilan diamond yang paling dekat dengan base bot. Pendekatan ini mengutamakan efisiensi pengumpulan dan penyimpanan diamond untuk meminimalkan risiko kehilangan akibat tackle musuh. Alternatif yang dieksplorasi meliputi pemilihan diamond bernilai tinggi namun tetap dekat dengan base, serta pemantauan kapasitas inventory agar bot dapat segera kembali menyimpan jika kapasitas penuh. Dengan mempertimbangkan nilai, jarak ke base, dan kondisi inventory, strategi ini memastikan alur pergerakan bot menjadi cepat, aman, dan efektif dalam meraih poin maksimal selama permainan.

3.2.5 Alternatif Greedy All In Diamond

Strategi greedy yang digunakan oleh bot ini mengombinasikan pemilihan target diamond dengan pendekatan blok berbasis nilai dan jarak, prioritas kembali ke base saat kapasitas penuh, serta penghindaran rintangan di jalur pergerakan. Bot membagi papan menjadi beberapa blok untuk mengidentifikasi area dengan potensi keuntungan terbesar dan memilih diamond secara lokal berdasarkan rasio nilai terhadap jarak, sehingga mengoptimalkan efisiensi pengumpulan. Selain itu, bot juga secara dinamis menyesuaikan tujuan saat menghadapi hambatan dengan jalur alternatif yang aman. Pendekatan ini menggunakan alternatif solusi greedy dengan memprioritaskan keuntungan maksimal sambil mempertimbangkan faktor jarak dan penghindaran risiko, sehingga bot dapat mengumpulkan diamond secara optimal dalam kondisi papan yang berubah-ubah.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

3.3.1 Alternatif Greedy by Shortest Distance Concerning Highest Diamond Value

a. Analisis Efisiensi Solusi

Algoritma greedy ini memiliki keunggulan efisiensi karena proses pengambilan keputusan dilakukan secara cepat, hanya dengan memilih kandidat terbaik berdasarkan kriteria tertentu seperti nilai diamond, jarak, atau rasio nilai terhadap jarak. Cara ini tidak perlu mencari semua kemungkinan, sehingga prosesnya lebih cepat dan cocok untuk permainan yang harus berjalan real-time. Namun, kecepatan ini tergantung pada seberapa banyak kandidat yang harus dipertimbangkan dan seberapa cepat fungsi pemilihan bisa berjalan.

b. Analisis Efektivitas Solusi

Solusi greedy efektif dalam mengumpulkan diamond dengan nilai tinggi secara cepat karena fokus pada keuntungan maksimum di setiap langkah. Strategi ini juga adaptif, seperti dengan mempertimbangkan kapasitas inventory dan memanfaatkan fitur teleporter untuk mempercepat pergerakan. Namun, karena greedy hanya melihat solusi lokal terbaik, strategi ini belum tentu menghasilkan solusi optimal secara global.

3.3.2 Alternatif Greedy by Surrounding Area

a. Analisis Efisiensi Solusi

Pendekatan greedy berdasarkan area sekitar ini cukup cepat karena bot hanya mencari diamond di area kecil yang sudah dibagi menjadi blok-blok 3x3 dan 5x5. Bot menghitung nilai total diamond dan jaraknya ke bot dengan cara yang mudah, sehingga tidak butuh waktu lama. Walaupun bot perlu mengecek beberapa blok, cara ini tetap lebih cepat daripada harus melihat semua posisi diamond di papan. Selain itu, pengecekan kelayakan berdasarkan kapasitas inventory juga sederhana dan tidak memerlukan waktu komputasi besar. Meski begitu, perhitungan beberapa blok dan jarak secara berulang bisa menambah beban, terutama jika jumlah diamond cukup banyak. Namun secara keseluruhan, metode ini tetap lebih cepat dibanding eksplorasi seluruh papan.

b. Analisis Efektivitas Solusi

Strategi ini bagus untuk mengumpulkan banyak diamond sekaligus karena bot fokus pada satu blok dengan nilai tertinggi dulu sebelum pindah ke blok lain. Dengan begitu, bot tidak perlu bolak-balik ke tempat yang berjauhan dan bisa mengumpulkan diamond dengan lebih efisien. Namun, kalau diamond tersebar jauh-jauh, strategi ini kurang efektif karena bot harus bergerak lebih jauh.

3.3.3 Alternatif Greedy by Attack Enemy

a. Analisis Efisiensi Solusi

Strategi ini termasuk yang standar karena tidak bisa mendapatkan nilai diamond yang stabil di setiap permainan. Namun bot tidak perlu mengecek semua diamond di peta. Bot hanya fokus pada base musuh dan memantau musuh yang sedang membawa banyak diamond. Proses pengambilan keputusannya jadi lebih sederhana dan hemat waktu. Namun, bot tetap harus membandingkan jarak ke base musuh dan posisi musuh, tapi ini tidak memakan banyak waktu karena hanya dilakukan untuk beberapa musuh saja.

b. Analisis Efektivitas Solusi

Strategi ini efektif kalau musuh sering membawa banyak diamond dan sedang dalam perjalanan ke base. Bot bisa menunggu di base musuh dan menyerang saat

mereka datang, lalu mengambil semua diamond sekaligus. Ini jauh lebih cepat dibanding harus mengumpulkan satu per satu. Selain itu, saat musuh terkena tackle, mereka harus respawn, dan itu membuat mereka kehilangan waktu. Tapi strategi ini kurang efektif kalau musuh jarang membawa diamond atau posisinya terlalu jauh, karena bot bisa menunggu lama tanpa hasil.

3.3.4 Alternatif Greedy by Closest to Base

a. Analisis Efisiensi Solusi

Strategi ini cukup efisien karena hanya perlu mencari diamond yang paling dekat dengan base, bukan seluruh diamond yang tersebar di peta. Proses seleksinya juga cepat karena hanya membandingkan jarak antara diamond dan base, lalu memilih yang terdekat. Strategi ini juga bisa menjadi tidak efisien jika diamond tersebar jauh dari base dan jumlah diamond di pertarungan itu sedikit.

b. Analisis Efektivitas Solusi

Strategi ini cukup efektif untuk menjaga keamanan diamond yang sudah dikumpulkan. Karena jaraknya dekat ke base, bot bisa cepat menyimpan diamond sebelum ditackle musuh. Ini mengurangi risiko kehilangan poin. Strategi ini juga cocok jika base bot berada di posisi strategis yang sering muncul diamond di sekitarnya. Namun, jika diamond muncul jauh dari base, strategi ini bisa membuat bot terlalu pasif dan kehilangan kesempatan untuk mendapatkan diamond yang lebih bernilai di tempat lain.

3.3.5 Alternatif Greedy All In Diamond

a. Analisis Efisiensi Solusi

Strategi ini termasuk efisien karena menggunakan pendekatan lokal, yaitu memilih diamond dalam blok 3x3 yang ada di sekitar bot, memilih diamond dengan nilai terbesar jika ada di dekatnya, prioritas kembali ke base saat kapasitas penuh, serta penghindaran rintangan di jalur pergerakan. Proses pemilihan hanya dilakukan pada area terbatas, sehingga perhitungan lebih cepat dan tidak membebani sistem permainan. Selain itu, strategi ini juga memperhitungkan rintangan, jadi bot tidak

buang waktu mencoba jalur yang tidak bisa dilewati. Ketika inventory sudah penuh, bot langsung kembali ke base, menghindari langkah-langkah yang tidak perlu.

- b. Strategi ini efektif dalam mengumpulkan diamond secara maksimal di setiap perjalanan. Bot hanya akan mengambil diamond yang bernilai tinggi dan berada di jalur yang aman. Pendekatan ini juga fleksibel karena bot bisa menyesuaikan jalur jika ada rintangan atau perubahan di peta permainan.

3.4 Strategi Greedy yang Dipilih

Sesuai dengan urutan penempatan penjelasan untuk tiap alternatif serta penjelasan diawal dimana dijelaskan bahwa setiap alternatif diatas merupakan versi bot greedy sehingga tentu saja konsep yang akan kami gunakan yaitu *Alternatif greedy All In Diamond*. Kami memilih bot ini karena kami merasa bahwa konsep ini sudah cukup bagus karena merupakan gabungan dari beberapa algoritma diatas.

Secara umum, kami tidak mengatakan bahwa pendekatan pendekatan versi sebelumnya buruk. Hal ini disebabkan karena fitur dalam permainan dapat terus diganti sehingga masing masing versi dapat memberikan kemampuan terbaik pada pendekatannya. Namun, kami memilih bot berdasarkan konsistensi yang ada.

Beberapa pendapat kami untuk setiap algoritma di atas :

- a. Alternatif Greedy by Shortest Distance Concerning Highest Diamond Value, Efisien karena hanya mempertimbangkan jarak, nilai, atau rasionya, cocok untuk game real-time. Namun, bisa melambat jika banyak diamond dan perhitungan rasio dilakukan terus-menerus.
- b. Alternatif Greedy by Surrounding Area, Cukup efisien karena fokus pada blok 3x3 atau 5x5 di sekitar bot. Dalam kondisi ini, strategi bisa menjadi kurang efisien, meskipun masih lebih baik dibanding pencarian global seluruh papan dan bisa melambat jika harus mengecek banyak blok sekaligus
- c. Alternatif Greedy by Attack Enemy, kurang efisien karena hanya fokus pada musuh dan base mereka dan jika musuh jarang membawa diamond atau terlalu jauh, sehingga bot bisa membuang waktu.

- d. Greedy by Shortest to Base sangat mengandalkan kondisi map sehingga sangat *coinflip* dalam implementasinya. Namun jika optimal dapat memberikan hasil yang bagus.
- e. Alternatif Greedy All In Diamond, Sangat efisien karena fokus lokal, cepat dalam evaluasi, adaptif terhadap perubahan peta, dan memperhitungkan rintangan. Hampir tidak ada kelemahan efisiensi signifikan.

Berikut merupakan hasil uji coba pencarian Alternatif Greedy All In Diamond secara solo dalam 5 percobaan dan dihitung rata rata jumlah diamond yang ada dalam map dalam waktu 60 detik :

Percobaan	Diamonds yang diperoleh
1	10
2	10
3	15
4	9
5	15
Rata-rata	11,8

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

4.1.1 Kode

```
from typing import Optional, List
from game.logic.base import BaseLogic
from game.models import Board, GameObject, Position
from game.util import get_direction

class CTABot(BaseLogic):
    static_goals: List[Position] = []
    static_temp_goal: Optional[Position] = None

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

    def next_move(self, bot: GameObject, board: Board):
        self.bot = bot
        self.board = board
        self.diamonds = board.diamonds
        self.base = bot.properties.base
        self.diamond_count = bot.properties.diamonds
        self.teleporters =
self.get_objects_by_type("TeleportGameObject")
self.red_buttons = self.get_objects_by_type("DiamondButtonGameObject")
        self.red_diamonds = [d for d in self.diamonds if
d.properties.points == 2]

        if bot.position in self.static_goals:
            self.static_goals.remove(bot.position)

        if bot.position == self.static_temp_goal:
            self.static_temp_goal = None

        # Prioritas: kembali ke base jika sudah 5 diamond
        if self.diamond_count >= 5:
            self.goal_position = self.base
            self.static_goals.clear()
        elif self.static_temp_goal:
```



```

        self.goal_position = self.static_temp_goal
    else:
        if not self.static_goals:
            self.evaluate_map_blocks()
            self.goal_position = self.get_closest(self.static_goals)

        if self.goal_position:
            dx, dy = get_direction(bot.position.x, bot.position.y,
self.goal_position.x, self.goal_position.y)
            self.check_obstacle_path(bot.position, self.goal_position)
        else:
            dx, dy = self.directions[self.current_direction]
            self.current_direction = (self.current_direction + 1) % 4

    return dx, dy

def get_objects_by_type(self, obj_type: str):
    return [obj for obj in self.board.game_objects if obj.type ==
obj_type]

def get_closest(self, positions: List[Position]) ->
Optional[Position]:
    current = self.bot.position
    return min(positions, key=lambda pos: abs(pos.x - current.x) +
abs(pos.y - current.y), default=None)

def evaluate_map_blocks(self):
    block_h, block_w = self.board.height // 3, self.board.width // 3
    blocks = [[[ for _ in range(3) for _ in range(3) ]
values = [[0 for _ in range(3) for _ in range(3) ]

    for diamond in self.diamonds:
        if diamond.properties.points == 2 and self.diamond_count == 4:
            continue
            i = diamond.position.y // block_h
            j = diamond.position.x // block_w
            blocks[i][j].append(diamond.position)
            values[i][j] += diamond.properties.points

    best_i, best_j, best_score = 0, 0, 0
    curr_i = self.bot.position.y // block_h
    curr_j = self.bot.position.x // block_w

    for i in range(3):
        for j in range(3):

```

```

        if values[i][j] == 0:
            continue
        distance = abs(curr_i - i) + abs(curr_j - j)
        score = values[i][j] + max(0, 5 - distance)
        if score > best_score:
            best_i, best_j, best_score = i, j, score

    if best_score == 0 and self.red_buttons:
        self.static_goals = [self.red_buttons[0].position]
    else:
        self.static_goals = blocks[best_i][best_j]

    def check_obstacle_path(self, start: Position, goal: Position):
        for obj_list in [self.teleporters, self.red_buttons,
self.red_diamonds]:
            for obj in obj_list:
                obs = obj.position
                if self.on_path(start, goal, obs):
                    new_goal = self.get_safe_detour(start, goal, obs)
                    if new_goal:
                        self.goal_position = new_goal
                        self.static_temp_goal = new_goal
                    return

    def on_path(self, start: Position, goal: Position, obs: Position)
-> bool:
        # Simple straight-line detection
        if start.x == goal.x == obs.x and min(start.y, goal.y) < obs.y
< max(start.y, goal.y):
            return True
        if start.y == goal.y == obs.y and min(start.x, goal.x) < obs.x
< max(start.x, goal.x):
            return True
        return False

    def get_safe_detour(self, start: Position, goal: Position, obs:
Position) -> Optional[Position]:
        # Simple dodge strategy: try side step
        if start.x == goal.x:
            side_x = start.x + 1 if start.x < self.board.width - 1
else start.x - 1
            return Position(start.y, side_x)
        elif start.y == goal.y:
            side_y = start.y + 1 if start.y < self.board.height - 1
else start.y - 1

```

```
        return Position(side_y, start.x)
    return None
```

4.1.2 Penjelasan Alur Program

Berikut adalah implementasi algoritma Greedy yang dilakukan bot dalam bentuk python. Strategi diimplementasikan dalam file mybot.py

Kamus dalam mybot.py

```
from typing import Optional, List
from game.logic.base import BaseLogic
from game.models import Board, GameObject, Position
from game.util import get_direction
```

```
class CTABot(BaseLogic):
    static_goals: List[Position] = []
    static_temp_goal: Optional[Position] = None
```

Terdapat class CTABot, yang berisikan :

- static_goals merupakan daftar target permanen yang harus dikunjungi bot (diamond).
- Static_temp_goal yang berfungsi sebagai tujuan sementara jika jalur ke goal utama terhalang rintangan.

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

`__init__` Method :

- directions: 4 arah gerakan (kanan, bawah, kiri, atas).
- current_direction: indeks arah saat ini jika tidak ada tujuan.
- goal_position: posisi target yang sedang dituju bot.

```

def next_move(self, bot: GameObject, board: Board):
    self.bot = bot
    self.board = board
    self.diamonds = board.diamonds
    self.base = bot.properties.base
    self.diamond_count = bot.properties.diamonds
    self.teleporters = self.get_objects_by_type("TeleportGameObject")
    self.red_buttons = self.get_objects_by_type("DiamondButtonGameObject")
    self.red_diamonds = [d for d in self.diamonds if d.properties.points == 2]

    if bot.position in self.static_goals:
        self.static_goals.remove(bot.position)

    if bot.position == self.static_temp_goal:
        self.static_temp_goal = None

    # Prioritas: kembali ke base jika sudah 5 diamond
    if self.diamond_count >= 5:
        self.goal_position = self.base
        self.static_goals.clear()
    elif self.static_temp_goal:
        self.goal_position = self.static_temp_goal
    else:
        if not self.static_goals:
            self.evaluate_map_blocks()
            self.goal_position = self.get_closest(self.static_goals)

    if self.goal_position:
        dx, dy = get_direction(bot.position.x, bot.position.y, self.goal_position.x, self.goal_position.y)
        self.check_obstacle_path(bot.position, self.goal_position)
    else:
        dx, dy = self.directions[self.current_direction]
        self.current_direction = (self.current_direction + 1) % 4

    return dx, dy

```

Fungsi utama def next_move(self, bot: GameObject, board: Board):

1. Inisialisasi atribut lokal (bot, papan, diamond, dsb).
2. Hapus posisi dari static_goals jika sudah tercapai.
3. Jika bot sudah punya 5 diamond → kembali ke base.
4. Jika ada static_temp_goal (menghindari rintangan) → ke sana dulu.
5. Kalau belum ada goal → evaluasi blok peta dan pilih target terbaik.
6. Jika sudah ada goal_position, tentukan arah ke sana dan periksa rintangan.
7. Kalau tidak ada goal → gerak acak mengikuti rotasi arah (current_direction).

```

def get_closest(self, positions: List[Position]) -> Optional[Position]:
    current = self.bot.position
    return min(positions, key=lambda pos: abs(pos.x - current.x) + abs(pos.y - current.y), default=None)

```

Fungsi get_objects_by_type untuk mengambil semua objek dari papan berdasarkan jenisnya (misalnya: Teleport, Button).

```

def evaluate_map_blocks(self):
    block_h, block_w = self.board.height // 3, self.board.width // 3
    blocks = [[[ ] for _ in range(3)] for _ in range(3)]
    values = [[[0] for _ in range(3)] for _ in range(3)]

    for diamond in self.diamonds:
        if diamond.properties.points == 2 and self.diamond_count == 4:
            continue
        i = diamond.position.y // block_h
        j = diamond.position.x // block_w
        blocks[i][j].append(diamond.position)
        values[i][j] += diamond.properties.points

    best_i, best_j, best_score = 0, 0, 0
    curr_i = self.bot.position.y // block_h
    curr_j = self.bot.position.x // block_w

    for i in range(3):
        for j in range(3):
            if values[i][j] == 0:
                continue
            distance = abs(curr_i - i) + abs(curr_j - j)
            score = values[i][j] + max(0, 5 - distance)
            if score > best_score:
                best_i, best_j, best_score = i, j, score

    if best_score == 0 and self.red_buttons:
        self.static_goals = [self.red_buttons[0].position]
    else:
        self.static_goals = blocks[best_i][best_j]

```

Evaluate_map_blocks berfungsi untuk :

- Membagi peta menjadi 3x3 blok.
- Menghitung nilai total diamond di tiap blok.
- Memilih blok dengan nilai tertinggi yang juga tidak terlalu jauh dari posisi bot.
- Jika tidak ada diamond yang valid → menuju ke tombol merah (DiamondButtonGameObject).

```
def check_obstacle_path(self, start: Position, goal: Position):
    for obj_list in [self.teleporters, self.red_buttons, self.red_diamonds]:
        for obj in obj_list:
            obs = obj.position
            if self.on_path(start, goal, obs):
                new_goal = self.get_safe_detour(start, goal, obs)
                if new_goal:
                    self.goal_position = new_goal
                    self.static_temp_goal = new_goal
                return
```

Check_obstacle_path berfungsi untuk :

- Mengecek apakah ada objek (teleporter, tombol merah, diamond merah) yang menjadi penghalang di jalur lurus.
- Jika ada → gunakan **get_safe_detour** untuk cari jalur menghindar.
- Set detour sebagai **static_temp_goal**.

```
def on_path(self, start: Position, goal: Position, obs: Position) -> bool:
    # Simple straight-line detection
    if start.x == goal.x == obs.x and min(start.y, goal.y) < obs.y < max(start.y, goal.y):
        return True
    if start.y == goal.y == obs.y and min(start.x, goal.x) < obs.x < max(start.x, goal.x):
        return True
    return False
```

On_path berfungsi untuk :

- Mengecek apakah objek obs ada di jalur lurus dari start ke goal.
- Berlaku untuk jalur vertikal atau horizontal saja.

```
def get_safe_detour(self, start: Position, goal: Position, obs: Position) -> Optional[Position]:
    # Simple dodge strategy: try side step
    if start.x == goal.x:
        side_x = start.x + 1 if start.x < self.board.width - 1 else start.x - 1
        return Position(start.y, side_x)
    elif start.y == goal.y:
        side_y = start.y + 1 if start.y < self.board.height - 1 else start.y - 1
        return Position(side_y, start.x)
    return None
```

Get_safe_detour berfungsi untuk :

- Jika jalur terhalang, fungsi ini memberi alternatif posisi menyamping (kiri/kanan atau atas/bawah).
- Bertujuan untuk menghindari penghalang tanpa rute kompleks.

4.2 Struktur Data yang Digunakan

Struktur data program yang kami buat untuk bot permainan Diamonds kami adalah sebagai berikut :

```
.
├── src/
│   ├── game/
│   │   ├── _pycache_
│   │   ├── logic/
│   │   │   ├── _init_.py
│   │   │   ├── base.py
│   │   │   ├── ctabot.py
│   │   │   ├── mybot.py
│   │   │   └── random.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
```

Seperti yang bisa dilihat pada struktur program diatas, terdapat 2 folder utama yang dijadikan modul dalam program, meliputi game dan logic Folder game berisi dengan file Python yang mengimplementasikan sistem keberjalanan game secara keseluruhan. Folder logic berisi program utama yang mengimplementasikan strategi greedy sebagai logika dari bot yang kami buat, tepatnya strategi yang digunakan terletak dalam file ctabot.py.

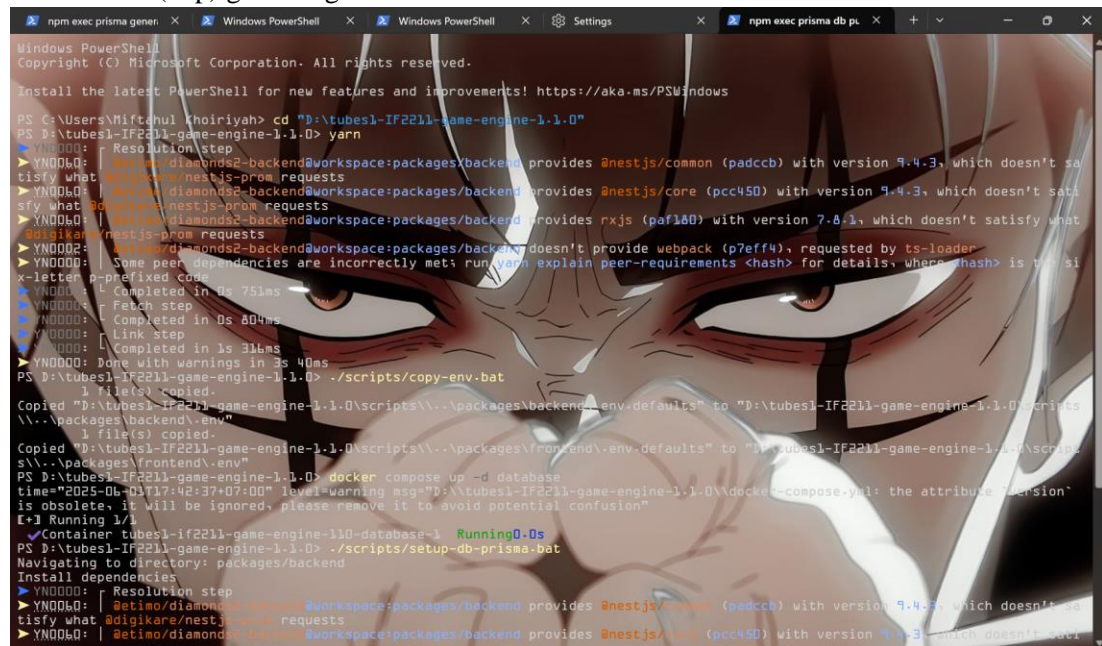
4.3 Pengujian Program

4.3.1 Skenario Pengujian

Pengujian bot AI dilakukan dalam lingkungan lokal server Etimo Bot. Bot akan dijalankan sebagai container Docker yang berfungsi sebagai klien logika permainan dan berkomunikasi langsung dengan game engine Etimo melalui protokol HTTP dalam jaringan Docker Compose. Selama permainan berlangsung, game engine akan mengirim permintaan ke endpoint bot (/move) untuk meminta keputusan langkah berdasarkan kondisi permainan yang diberikan. Bot kemudian memproses data tersebut menggunakan algoritma yang telah ditanamkan dan merespons dengan arah gerakan yang dipilih. Dengan pendekatan ini, lingkungan eksekusi bot menjadi terisolasi, portabel, dan konsisten di berbagai sistem, serta mendukung pengujian paralel dan otomatis melalui container orchestration.

Langkah Pengujian :

1. Lakukan instalasi dan konfigurasi awal dengan mendownload docker desktop, [Node.js](#), dan source code (.zip) game engine.

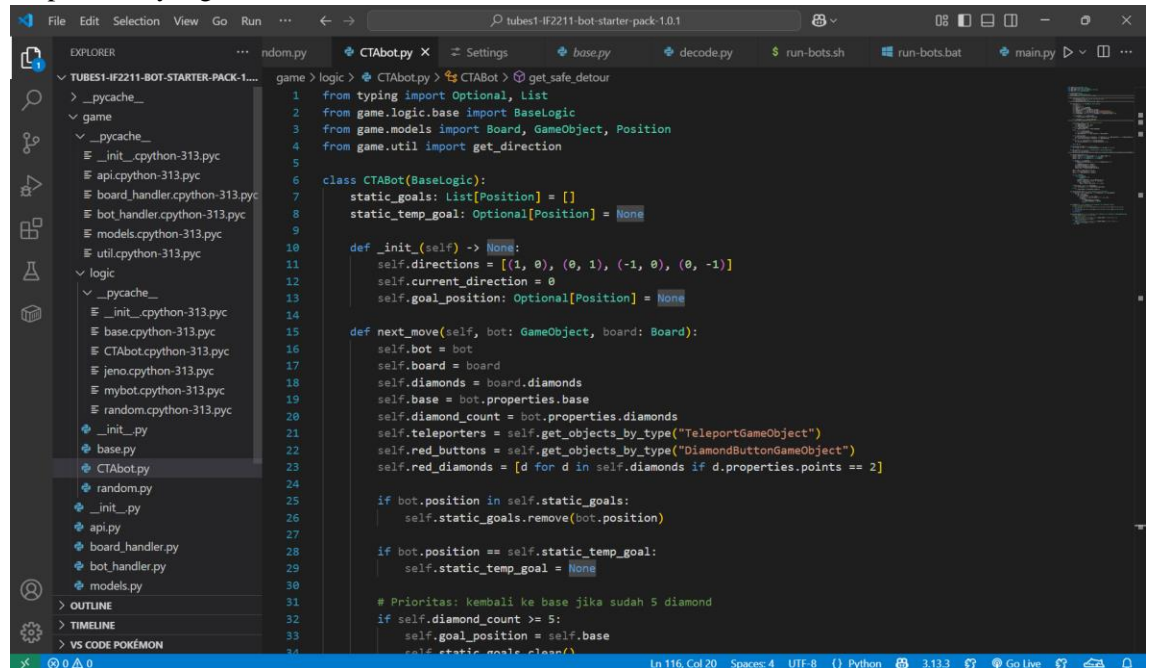


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Miftahul Khoiriyah> cd "D:\tubes1-IF2211-game-engine-1.1.0"
PS D:\tubes1-IF2211-game-engine-1.1.0> yarn
YN0000: Resolution step
YN0000: @etimo/diamonds2-backend@workspace:packages/backend provides @nestjs/common (padccb) with version 9.4.3, which doesn't satisfy what @adigikare/nestjs-prom requests
YN0000: @etimo/diamonds2-backend@workspace:packages/backend provides @nestjs/core (pcc450) with version 9.4.3, which doesn't satisfy what @adigikare/nestjs-prom requests
YN0000: @etimo/diamonds2-backend@workspace:packages/backend provides rxjs (paf100) with version 7.8.1, which doesn't satisfy what @adigikare/nestjs-prom requests
YN0002: @etimo/diamonds2-backend@workspace:packages/backend doesn't provide webpack (p7eff4), requested by ts-loader
YN0000: Some peer dependencies are incorrectly met; run yarn explain-peer-requirements <hash> for details, where <hash> is the six-letter a-prefixed code
YN0000: Completed in 0s 751ms
YN0000: Fetch step
YN0000: Completed in 0s 804ms
YN0000: Link step
YN0000: Completed in 1s 311ms
YN0000: Done with warnings in 3s 40ms
PS D:\tubes1-IF2211-game-engine-1.1.0> ./scripts/copy-env.bat
Copied "D:\tubes1-IF2211-game-engine-1.1.0\scripts\...\packages\backend\env.defaults" to "D:\tubes1-IF2211-game-engine-1.1.0\scripts\...\packages\backend\env"
Copied "D:\tubes1-IF2211-game-engine-1.1.0\scripts\...\packages\frontend\env.defaults" to "D:\tubes1-IF2211-game-engine-1.1.0\scripts\...\packages\frontend\env"
PS D:\tubes1-IF2211-game-engine-1.1.0> docker compose up -d database
time="2025-06-01T17:42:37+07:00" level=warning msg="D:\tubes1-IF2211-game-engine-1.1.0\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[*] Running 1/1
Container tubes1-if2211-game-engine-110-database-1 Running 0.0s
PS D:\tubes1-IF2211-game-engine-1.1.0> ./scripts/setup-db-prisma.bat
Navigating to directory: packages/backend
Installing dependencies
YN0000: Resolution step
YN0000: @etimo/diamonds2-backend@workspace:packages/backend provides @nestjs/common (padccb) with version 9.4.3, which doesn't satisfy what @adigikare/nestjs-prom requests
YN0000: @etimo/diamonds2-backend@workspace:packages/backend provides @nestjs/core (pcc450) with version 9.4.3, which doesn't satisfy what @adigikare/nestjs-prom requests
```

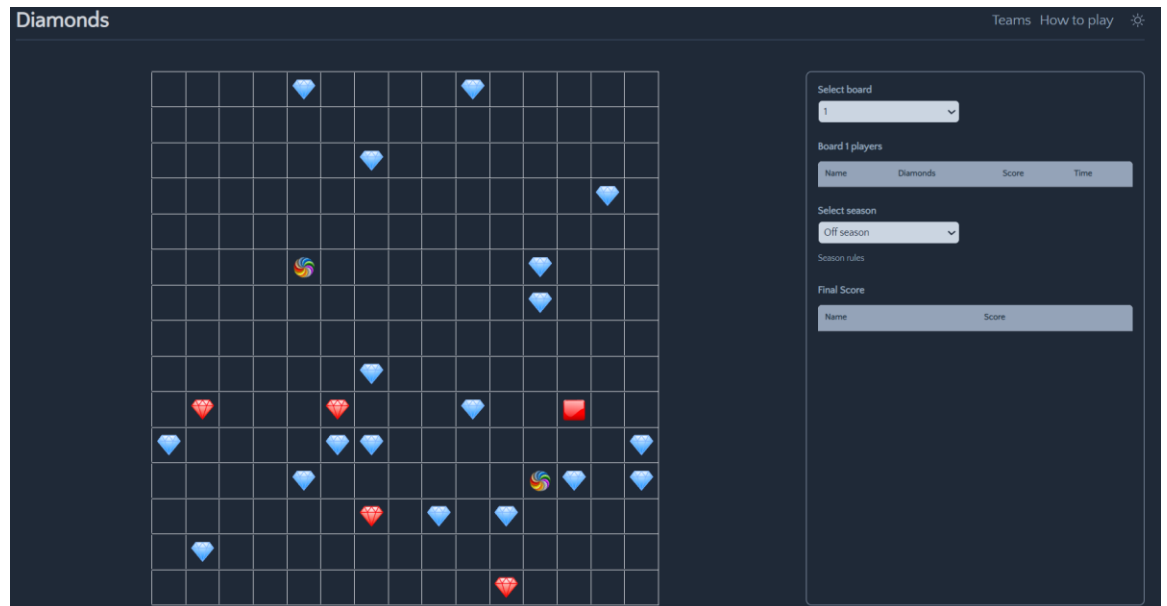

2. Siapkan bot yang telah di buat.



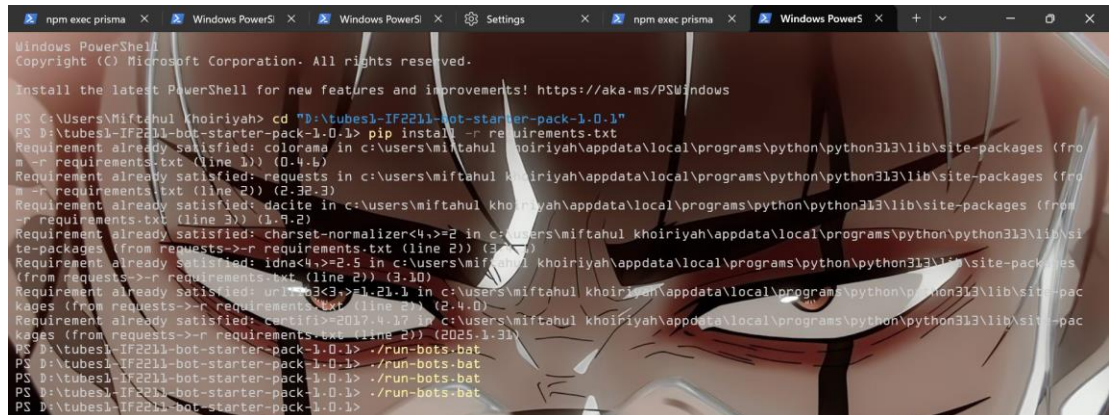
```
game > logic > CTABot.py > CTABot > get_safe_detour
1 from typing import Optional, List
2 from game.logic.base import BaseLogic
3 from game.models import Board, GameObject, Position
4 from game.util import get_direction
5
6 class CTABot(BaseLogic):
7     static_goals: List[Position] = []
8     static_temp_goal: Optional[Position] = None
9
10    def __init__(self) -> None:
11        self.directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
12        self.current_direction = 0
13        self.goal_position: Optional[Position] = None
14
15    def next_move(self, bot: GameObject, board: Board):
16        self.bot = bot
17        self.board = board
18        self.diamonds = board.diamonds
19        self.base = bot.properties.base
20        self.diamond_count = bot.properties.diamonds
21        self.teleporters = self.get_objects_by_type("TeleportGameObject")
22        self.red_buttons = self.get_objects_by_type("DiamondButtonGameObject")
23        self.red_diamonds = [d for d in self.diamonds if d.properties.points == 2]
24
25        if bot.position in self.static_goals:
26            self.static_goals.remove(bot.position)
27
28        if bot.position == self.static_temp_goal:
29            self.static_temp_goal = None
30
31        # Prioritas: kembali ke base jika sudah 5 diamond
32        if self.diamond_count >= 5:
33            self.goal_position = self.base
34            self.static_goals.clear()
```

*pada percobaan kali ini kami menggunakan bahasa python untuk membuat bot etimo game.

3. Kunjungi *frontend* melalui <http://localhost:8082/>. Berikut adalah tampilan awal *frontend*.



4. Run Bot melalui terminal/powershell menggunakan (./run-bots.bat)

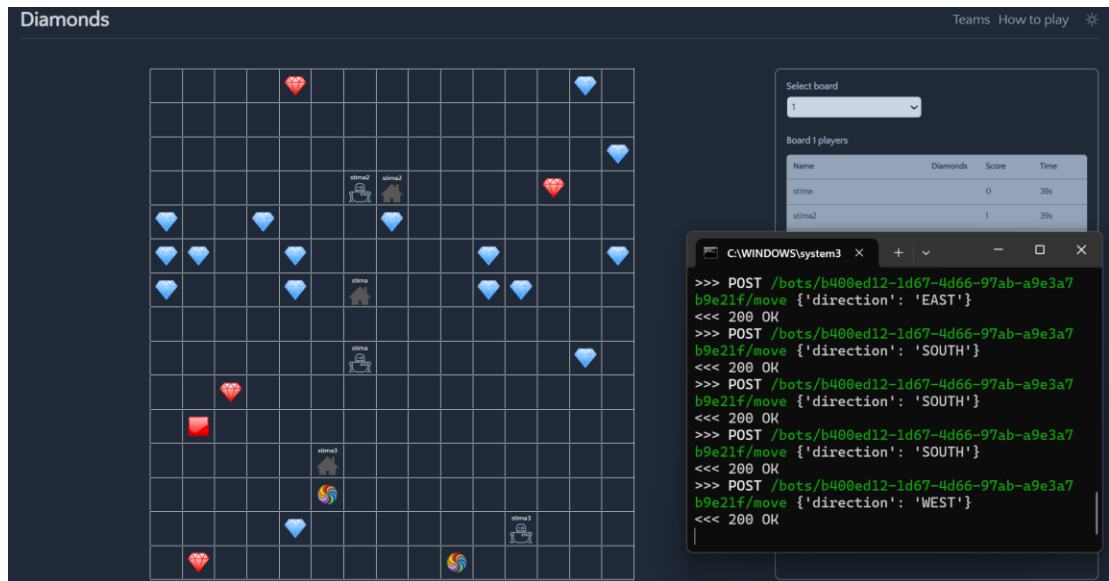


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Miftahul Khoiriyah> cd "D:\tubes1-IF2211-bot-starter-pack-1.0.1"
PS D:\tubes1-IF2211-bot-starter-pack-1.0.1> pip install -r requirements.txt
Requirement already satisfied: colorama in c:\users\miftahul khoiriyah\appdata\local\programs\python\python313\lib\site-packages (from
-r requirements.txt (line 1)) (0.4.6)
Requirement already satisfied: requests in c:\users\miftahul khoiriyah\appdata\local\programs\python\python313\lib\site-packages (from
-r requirements.txt (line 2)) (2.32.3)
Requirement already satisfied: dacite in c:\users\miftahul khoiriyah\appdata\local\programs\python\python313\lib\site-packages (from
-r requirements.txt (line 3)) (1.9.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\miftahul khoiriyah\appdata\local\programs\python\python313\lib\si
te-packages (from requests->r requirements.txt (line 2)) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\miftahul khoiriyah\appdata\local\programs\python\python313\lib\site-packag
es (from requests->r requirements.txt (line 2)) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\miftahul khoiriyah\appdata\local\programs\python\python313\lib\site-pac
kages (from requests->r requirements.txt (line 2)) (2.4.0)
Requirement already satisfied: aiohttp<4.0.0, >=3.9.2 in c:\users\miftahul khoiriyah\appdata\local\programs\python\python313\lib\site-pac
kages (from requests->r requirements.txt (line 2)) (2025.11.3)
PS D:\tubes1-IF2211-bot-starter-pack-1.0.1> ./run-bots.bat
PS D:\tubes1-IF2211-bot-starter-pack-1.0.1> ./run-bots.bat
PS D:\tubes1-IF2211-bot-starter-pack-1.0.1> ./run-bots.bat
PS D:\tubes1-IF2211-bot-starter-pack-1.0.1> ./run-bots.bat
```

5. Etimo Bot berhasil dijalankan.



Skenario Pengujian Etimo Bot.

- Skenario 1 : Tes Kemampuan Pencarian Clustering

Tujuan : Menguji kemampuan bot dalam mengenali dan memprioritaskan area pada peta permainan yang memiliki konsentrasi (clustering) diamond tertinggi sebagai target utama pencarian.

Deskripsi : Bot dapat menganalisis distribusi diamond yang tersebar di seluruh peta dan mengidentifikasi area-area dengan kepadatan diamond yang tinggi. Pengambilan keputusan diarahkan agar bot lebih fokus mengunjungi blok atau wilayah dengan nilai kumulatif diamond tertinggi dibanding wilayah lain, untuk memaksimalkan efisiensi pengumpulan.

Indikator Keberhasilan : Bot memprioritaskan masuk ke blok dengan total nilai diamond tertinggi, bot mampu menghindari wilayah dengan nilai rendah saat terdapat alternatif cluster bernilai tinggi, serta perilaku bot konsisten terhadap variasi distribusi diamond pada peta.

- Skenario 2 : Tes Kemampuan Kembali ke Base

Tujuan : Menguji kemampuan bot dalam mengenali waktu yang tepat untuk kembali ke base dan memilih jalur optimal agar proses kembali berjalan efisien, terutama setelah berhasil mengumpulkan 5 diamond.

Deskripsi : Bot memiliki batasan kapasitas maksimal 5 diamond yang mengharuskan bot untuk kembali ke base. Hal ini mengukur kemampuan bot dalam :

1. Mengenali kondisi pengembalian
2. Memilih jalur tercepat dan teraman kembali ke base, mempertimbangkan posisi bot, dan hambatan lain.
3. Menghindari area berisiko tinggi (bertabrakan dengan bot lain) yang dapat mengambil diamond bot.

Indikator keberhasilan : Bot mampu mendeteksi kondisi yang memicu perintah kembali ke base (maksimal perolehan diamond 5), Bot memilih jalur tercepat dengan prioritas keamanan yang tinggi, Bot konsisten dalam mengenali kondisi pengembalian dan merespons dengan jalur yang efisien di berbagai variasi distribusi diamond dan hambatan peta.

- Skenario 3 : Tes Kemampuan fokus ke daerah dengan densitas diamond tertinggi jika jarak dekat

Tujuan : Menguji kemampuan bot dalam memprioritaskan area dengan konsentrasi diamond tertinggi yang berada pada jarak dekat dari posisi bot saat ini. Fokus ini dimaksudkan untuk memaksimalkan pengumpulan diamond tanpa membuang waktu ke area yang jauh meskipun bernilai tinggi.

Deskripsi : Bot akan memindai area di sekitar posisinya saat ini untuk mendeteksi konsentrasi diamond terdekat. Ketika mendeteksi area dengan densitas diamond yang tinggi pada radius jarak dekat, bot akan memprioritaskan untuk masuk dan memaksimalkan pengumpulan diamond di area tersebut. Jika tidak ada area dekat dengan densitas tinggi, bot akan melanjutkan pergerakan normal atau mencari area lain dengan kriteria yang lebih longgar.

Indikator keberhasilan : Bot mendeteksi dan memprioritaskan area dengan densitas diamond tertinggi pada jarak dekat, bot dapat mengabaikan area yang lebih jauh meskipun memiliki nilai lebih besar jika ada area dekat yang cukup menguntungkan, bot mampu beradaptasi dengan peta yang berubah-ubah, serta bot memutuskan pergerakan menuju area dekat bernilai tinggi secara konsisten dan cepat.

- Skenario 4 : Tes Kemampuan fokus ke daerah dengan densitas diamond poin 1 jika jarak dekat

Tujuan : Menguji kemampuan bot dalam memprioritaskan area dengan konsentrasi diamond poin 1 jika berada pada jarak dekat dari posisi bot saat ini. Fokus ini dimaksudkan untuk memaksimalkan pengumpulan diamond tanpa membuang waktu ke area yang lebih jauh atau area dengan jenis diamond lain yang nilainya lebih tinggi.

Deskripsi : Bot akan memindai area di sekitar posisinya saat ini untuk mendeteksi area dengan densitas diamond point 1 (diamond dengan nilai dasar). Ketika mendeteksi area dengan konsentrasi diamond poin 1 yang tinggi dalam radius jarak dekat, bot akan memprioritaskan untuk masuk dan memaksimalkan pengumpulan diamond tersebut. Jika tidak ada area dekat dengan densitas tinggi diamond point 1, bot akan melanjutkan pergerakan normal atau mencari area lain dengan kriteria yang lebih longgar, misalnya area dengan diamond jenis lain atau area dengan distribusi campuran.

Indikator Keberhasilan : Bot mendeteksi dan memprioritaskan area dengan densitas diamond poin 1 tertinggi pada jarak dekat, bot dapat mengabaikan area lain yang lebih jauh meskipun memiliki diamond dengan poin lebih tinggi, bot mampu beradaptasi dengan distribusi diamond yang berubah-ubah di peta, serta bot memutuskan pergerakan menuju area dengan diamond poin 1 secara konsisten dan cepat.

- Skenario 5 : Tes kemampuan menggunakan portal jika menguntungkan

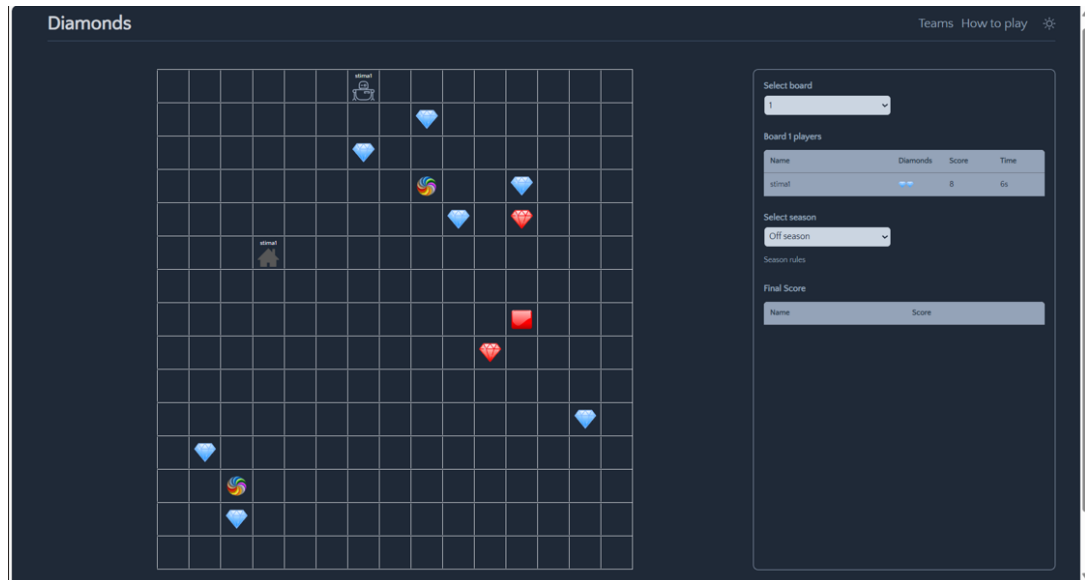
Tujuan : Menguji kemampuan bot dalam mengenali keberadaan portal dan memanfaatkannya jika portal tersebut memberikan keuntungan strategis, seperti mempersingkat perjalanan ke area dengan diamond bernilai tinggi, menghemat waktu kembali ke base, atau menghindari area berisiko.

Deskripsi : Bot akan memindai peta untuk mendeteksi portal yang berada di sekitar posisinya saat ini. Ketika mendeteksi portal yang berada dalam radius jarak dekat dan menganalisis bahwa portal tersebut menguntungkan (misalnya mempersingkat perjalanan ke area dengan konsentrasi diamond tinggi atau memotong jarak ke base), bot akan memutuskan untuk masuk portal tersebut. Jika portal tidak memberikan keuntungan signifikan dibandingkan rute biasa, bot akan mengabaikan portal dan tetap menggunakan jalur konvensional. Fokus pengujian ini adalah untuk memastikan bot mampu menilai keuntungan penggunaan portal dibandingkan jalur lain dalam kondisi peta yang bervariasi.

Indikator Keberhasilan: Bot mendeteksi portal yang berada pada radius jarak dekat, bot memprioritaskan penggunaan portal jika memberikan keuntungan signifikan, bot mampu mengabaikan portal jika rute biasa lebih menguntungkan, bot mampu beradaptasi dengan kondisi peta yang dinamis (misalnya perubahan distribusi diamond atau penempatan portal), serta bot memutuskan pergerakan secara konsisten dan cepat berdasarkan analisis keuntungan penggunaan portal.

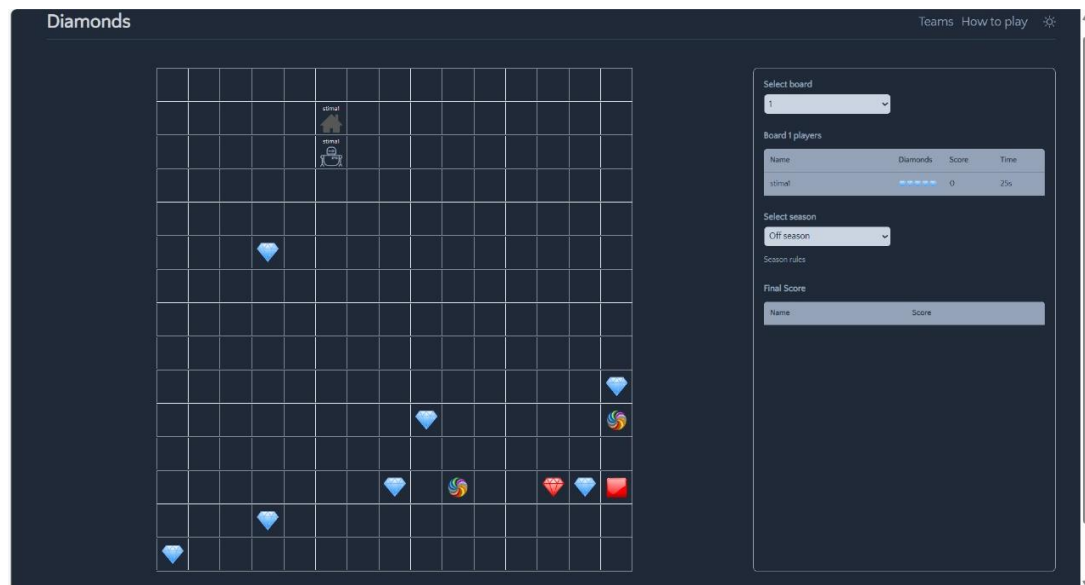
4.3.2 Hasil Pengujian dan Analisis

a. Tes Kemampuan Pencarian Clustering



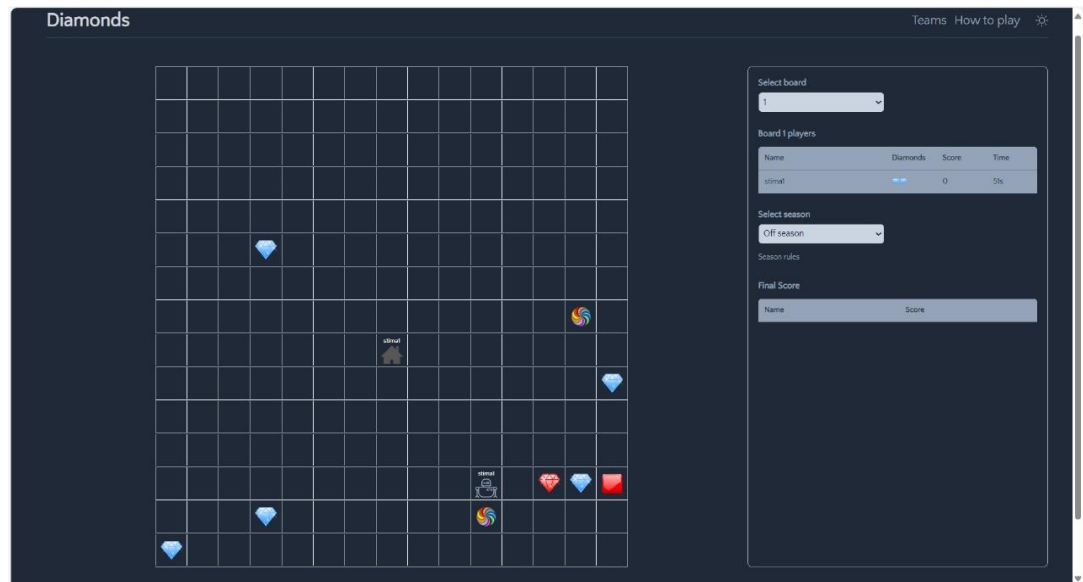
Map permainan dibagi menjadi beberapa blok 3x3, di mana beberapa blok berisi banyak diamond yang berdekatan (clustering), sedangkan lainnya hanya memiliki satu atau dua diamond yang tersebar. Bot diharapkan dapat menganalisis peta secara menyeluruh, menghitung nilai total dalam tiap blok, dan memilih untuk bergerak menuju blok yang memiliki nilai kumulatif tertinggi, bukan hanya berdasarkan jarak terdekat.

b. Tes Kemampuan Kembali ke Base



Bot mampu memantau jumlah diamond yang dibawa, dan saat mencapai jumlah maksimum, bot akan segera mengubah tujuan dari pengumpulan diamond ke navigasi menuju base, tanpa mengambil tambahan diamond di sekitar.

- c. Tes Kemampuan fokus ke daerah dengan densitas diamond tertinggi jika jarak dekat



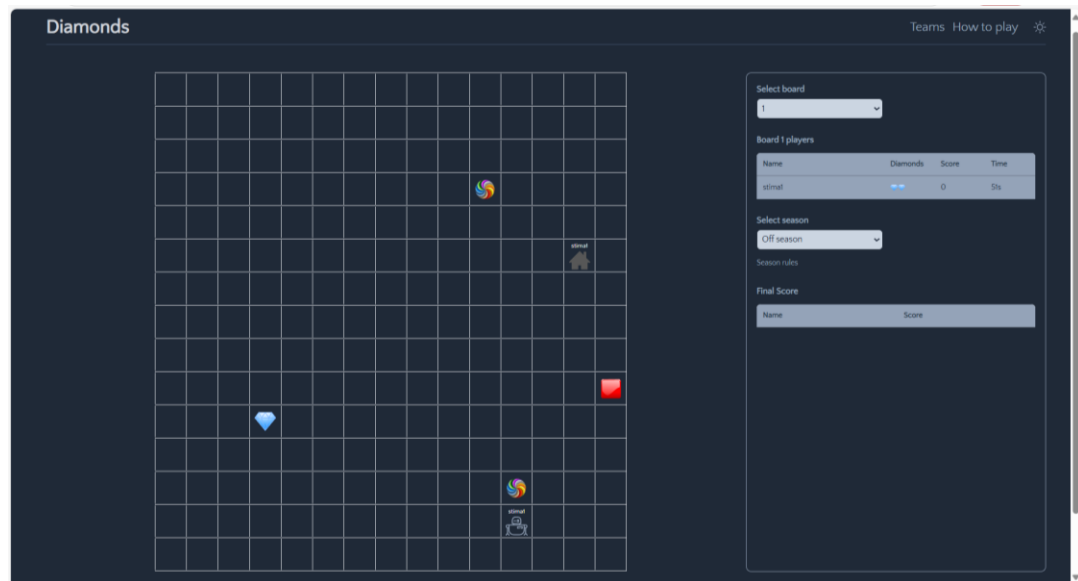
Bot mampu mengevaluasi sekelompok diamond yang berdekatan (klaster lokal) dan memilih untuk menuju ke area dengan konsentrasi diamond yang tinggi dalam radius tertentu. Hal ini bermanfaat untuk efisiensi gerak dan pengumpulan skor maksimum dalam jumlah langkah minimal.

- d. Tes Kemampuan fokus ke daerah dengan densitas diamond poin 1 jika jarak dekat



Bot mampu mengenali dan memprioritaskan area dengan konsentrasi (densitas) diamond tertinggi di sekitarnya, terutama jika berada dalam jarak dekat.

- e. Tes kemampuan menggunakan portal jika menguntungkan



Bot mampu mengevaluasi jalur yang tersedia dan memilih menggunakan portal apabila jarak total yang ditempuh melalui portal lebih pendek dibandingkan jalur biasa, terutama ketika mengejar diamond bernilai tinggi atau kembali ke base saat membawa 5 diamond.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dalam mengembangkan bot untuk permainan *Etimo Diamonds*, kami menerapkan strategi Greedy dan menemukan bahwa algoritma ini cukup efektif dalam mencapai solusi lokal maksimum berdasarkan kondisi permainan saat itu. Algoritma greedy tergolong metode optimasi yang baik untuk pengambilan keputusan karena solusi lokal yang dipilih sering kali mendekati solusi global. Namun, kelemahan utamanya adalah kemungkinan gagal menemukan solusi terbaik secara keseluruhan, terutama dalam kondisi permainan yang tidak menguntungkan, seperti posisi spawn yang jauh dari diamond. Oleh karena itu, pemilihan algoritma harus disesuaikan dengan kebutuhan dan dinamika permainan.

5.2 Saran

Selama mengerjakan tugas besar ini, kami menyadari pentingnya tetap fokus pada tujuan utama, yaitu menciptakan bot dengan efisiensi setinggi mungkin, terutama dalam hal pengambilan diamond. Namun, dalam prosesnya, kami beberapa kali terlalu terpaku pada aspek kompetitif permainan hingga melupakan konsep awal yang telah ditetapkan. Padahal, dalam permainan kompetitif, ada banyak faktor yang memengaruhi kemenangan dan tidak semuanya bergantung pada algoritma greedy semata. Karena itu, kami memastikan bahwa pemilihan strategi bot tetap mengacu pada prinsip utama kami, yaitu bot yang mampu mengumpulkan diamond sebanyak mungkin dalam waktu tertentu, sesuai dengan esensi dari algoritma greedy yang kami terapkan.

LAMPIRAN

- A. **Repository Github** (https://github.com/11-066-ElfaNoviana/Tubes1_CTA)
- B. **Video Penjelasan** (https://drive.google.com/drive/folders/1fRgubFnSi4sK-xEYZGgXUWLWisX_No_p)

DAFTAR PUSTAKA

- [1] J. Zhao, T. Huang, F. Pang and Y. Liu, "Genetic Algorithm Based on Greedy Strategy in the 0-1 Knapsack Problem," *Third International Conference on Genetic and Evolutionary Computing*, vol. 10.1109/WGEC.2009.43., pp. 105-107, 2009.
- [2] A. Taqwiym, "PENDEKATAN ALGORITMA GREEDY UNTUK MENENTUKAN LANGKAH BIDAK PADA PERMAINAN CHECKERS," *urnal Teknologi, Informasi Dan Industri*, vol. Vol 1, pp. 89-97, 2018.
- [3] S. Kristi, "Rancang bangun Game Finding Cockroach dengan Algoritma Greedy Best First Search untuk optimasi dalam menemukan posisi objek," *Diploma thesis, UIN Sunan Gunung Djati Bandung.*, vol. Vol 1, 2019.