

**LAPORAN TUGAS BESAR I
IF2211 STRATEGI ALGORITMA**

**PEMANFAATAN ALGORITMA GREEDY DALAM
PEMBUATAN BOT PERMAINAN DIAMONDS**



**KELOMPOK TONGGOKU NYALEG NDES
ANGGOTA:**

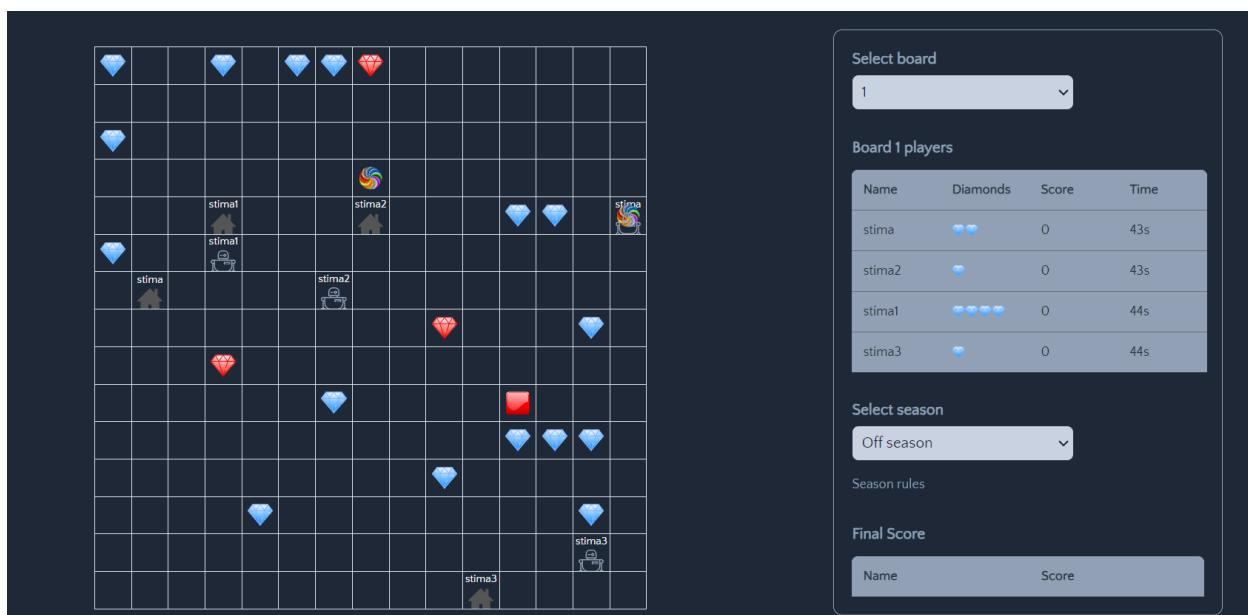
- 1. 13522004 EDUARDUS ALVITO KRISTIADI**
- 2. 13522016 ZACHARY SAMUEL TOBING**
- 3. 13522112 DIMAS BAGOES HENDRIANTO**

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024**

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.



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

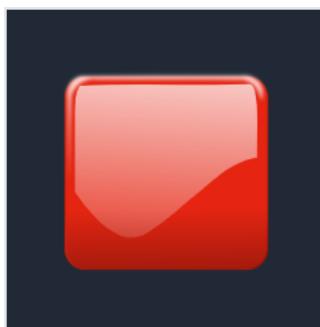
Komponen-komponen dari permainan Diamonds antara lain:

1. Diamonds



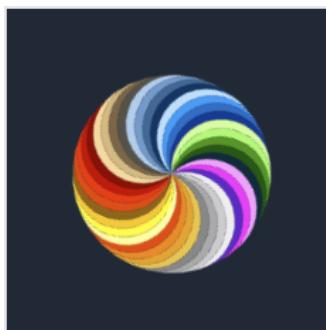
Untuk memenangkan pertandingan, kita harus mengumpulkan diamond ini sebanyak-banyaknya dengan melewati/melangkahinya. Terdapat 2 jenis diamond yaitu diamond biru dan diamond merah. Diamond merah bernilai 2 poin, sedangkan yang biru bernilai 1 poin. Diamond akan di-regenerate secara berkala dan rasio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

2. Red Button/Diamond Button



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

3. Teleporters



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

4. Bots and Bases



Pada game ini kita akan menggerakkan bot untuk mendapatkan diamond sebanyak banyaknya. Semua bot memiliki sebuah Base dimana Base ini akan digunakan untuk menyimpan diamond yang sedang dibawa. Apabila diamond disimpan ke base, score bot akan bertambah senilai diamond yang dibawa dan inventory (akan dijelaskan di bawah) bot menjadi kosong.

5. Inventory

Name	Diamonds	Score	Time
stima	♥♥	0	43s
stima2	♥	0	43s
stima1	♥♥♥♥	0	44s
stima3	♥	0	44s

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

BAB II

LANDASAN TEORI

2.1 Algoritma Greedy

Algoritma greedy adalah algoritma yang memecahkan persoalan secara langkah per langkah (step by step) dengan prinsip take what you can get now, yaitu mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan dan “berharap” bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global. Algoritma ini berusaha untuk mencapai solusi yang optimal dengan melakukan pemilihan yang paling rasional dan efektif pada setiap tahapnya.

2.2 Elemen-elemen Algoritma Greedy

1. Himpunan kandidat , C : berisi kandidat yang akan dipilih pada setiap langkah.
Contoh: simpul sisi di dalam graf , job, task, koin , benda , karakter , dsb
2. Himpunan solusi, S : berisi kandidat yang sudah dipilih.
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
4. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
5. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
6. Fungsi obyektif : memaksimumkan atau meminimumkan.

2.3 Game Engine Permainan Diamonds

Dalam rangka memulai menjalankan game dan mengimplementasikan kode bot, diperlukan starter pack permainan Diamonds. Starter pack dapat diunduh melalui link ini: <https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>. Ada beberapa komponen penting yang perlu diketahui untuk memahami cara kerja game, yakni:

1. Game engine: komponen yang berperan dalam mengimplementasikan logika dan peraturan game.
2. Bot starter pack: yang berfungsi memanggil API serta berisi bot logic yang mengatur strategi dan taktik dari pergerakan bot demi memenangkan permainan.

Cara melakukan setup pada game engine dan bot starter pack adalah sebagai berikut.

1. Install semua requirements dan prerequisite (Python, NodeJS, Docker desktop, Yarn, Game Engine, Bot Starter Pack)
2. Untuk menjalankan game engine, masuk ke root directory dari project.

3. Lakukan setup default environment variable dengan menjalankan script berikut. Untuk Windows: `./scripts/copy-env.bat`
4. Lakukan setup local database (buka aplikasi docker desktop terlebih dahulu, lalu jalankan command berikut di terminal) : `docker compose up -d database`
5. Jalankan script berikut untuk melakukan setup pada Prisma: `./scripts/setup-db-prisma.bat`
6. Jalankan website permainan Diamonds (`npm run build`) lalu (`npm run start`)
7. Untuk menjalankan bot starter pack, masuk ke root directory dari bot starter Pack.
8. Untuk menjalankan satu bot saja: `python main.py --logic Random --email=your_email@example.com --name=your_name --password=your_password --team etimo`
9. Apabila ingin menjalankan beberapa bot sekaligus dalam satu permainan: `./run-bots.bat`
10. Apabila ingin mengubah bot mana dan jumlah bot yang ingin ditampilkan dan diadu dalam permainan, modifikasilah bagian ini. Modifikasi dapat dilakukan pada nama bot, email, logic bot, dan password.

```
run-bots.bat
1  @echo off
2  start cmd /c "python main.py --logic Random --email=test@email.com --name=stima --password=123456 --team etimo"
3  start cmd /c "python main.py --logic Random --email=test1@email.com --name=stima1 --password=123456 --team etimo"
4  start cmd /c "python main.py --logic Random --email=test2@email.com --name=stima2 --password=123456 --team etimo"
5  start cmd /c "python main.py --logic Random --email=test3@email.com --name=stima3 --password=123456 --team etimo"
6
```

2.4 Alur Cara Kerja Program Game Diamonds

1. Inisialisasi: Kelas MyBot diinisialisasi dengan atribut untuk menyimpan arah, posisi tujuan, dan arah saat ini. Method `next_move` adalah tempat bot memutuskan aksi selanjutnya berdasarkan keadaan permainan yang direpresentasikan oleh `board` dan `board_bot` (bot itu sendiri adalah juga sebagai `GameObject`).
2. Proses Pengambilan Keputusan: Dalam `next_move`, bot mengevaluasi keadaan permainan untuk memutuskan jalannya aksi. Pengambilan keputusan adalah sebagai berikut:
 - Menekan Tombol: Jika total berlian biru dan merah di papan kurang dari 17, dan bot dekat dengan tombol (kurang dari 5 langkah) dengan kurang dari 4 berlian, bot akan mengutamakan menekan tombol untuk menghasilkan lebih banyak berlian.
 - Menyetorkan Berlian: Jika bot telah mengumpulkan 5 berlian atau jika sisa waktu permainan rendah (kurang dari 3 detik dari waktu yang dibutuhkan untuk kembali ke basis dengan lebih dari 0 berlian), bot akan mengutamakan kembali ke basis untuk menyetorkan berlian.
 - Mengumpulkan Berlian: Jika bot memiliki ruang untuk lebih banyak berlian (kurang dari 4), bot akan menggunakan fungsi `get_nearest_blue` atau `get_nearest_red` untuk menemukan berlian terdekat dari masing-masing kemudian memutuskan mana

yang akan dikejar berdasarkan perbandingan jarak. Jika sebuah berlian berada dalam tiga langkah, itu dapat mengesampingkan keputusan lain untuk segera mengumpulkannya.

- Kondisi Khusus: Juga termasuk logika untuk kondisi khusus seperti menghindari portal ketika sejajar dengan basis bot untuk mencegah perjalanan yang tidak diinginkan.
- 3. Implementasi Algoritma Greedy: Pendekatan greedy bot terlihat jelas dalam cara ia selalu mencari aksi bermanfaat terdekat (mengumpulkan berlian, menekan tombol, atau kembali ke basis) yang memberikan hasil dan keuntungan terbesar. Bot akan mengevaluasi setiap kemungkinan dari berbagai aksi berdasarkan keadaan permainan saat ini dan memilih yang tampaknya menawarkan nilai langsung tertinggi.
- 4. Eksekusi dan Pergerakan: Setelah menentukan posisi tujuan (baik itu berlian, tombol, atau basis), bot menghitung arah untuk bergerak menuju tujuan tersebut. Jika tidak ada tujuan spesifik yang ditetapkan, bot akan berkeliling berdasarkan pola yang telah ditentukan yang berubah secara acak dari waktu ke waktu.
- 5. Fungsi Utilitas: Program ini mencakup beberapa fungsi utilitas (get_nearest_blue, get_nearest_red, get_portal, dll.) yang membantu dalam proses pengambilan keputusan dengan menyediakan informasi tentang papan permainan, seperti berlian terdekat, lokasi portal, dan arah ke basis rumah.

2.5 Menjalankan Bot

Untuk menjalankan bot, biasanya akan memiliki loop permainan di mana setiap tick (atau frame) dari permainan, kami memanggil metode next_move dari instansi bot kami, menyampaikan keadaan saat ini dari bot dan papan permainan. Metode next_move kemudian mengembalikan arah di mana bot memutuskan untuk bergerak berdasarkan strateginya saat ini. Arah ini diterapkan untuk memperbarui posisi bot pada papan permainan untuk tick permainan berikutnya.

BAB III

APLIKASI STRATEGI GREEDY

3.1 Alternatif Greedy

3.1.1 Greedy by Exploration and Exploitation Balance

1. Mapping Elemen Greedy

- Himpunan Kandidat: Semua kemungkinan langkah untuk mengumpulkan diamonds atau menghindari musuh.
- Himpunan Solusi: Langkah yang mengoptimalkan pengumpulan diamonds sambil meminimalkan risiko terkena serangan.
- Fungsi Solusi: Memeriksa apakah langkah yang dipilih berhasil menyeimbangkan antara eksplorasi untuk diamonds dan eksploitasi dari posisi saat ini.
- Fungsi Seleksi: Memilih langkah yang memberikan rasio terbaik antara potensi keuntungan (mengumpulkan diamonds) dan potensi kerugian (terkena serangan).
- Fungsi Kelayakan: Memeriksa apakah langkah memenuhi kondisi permainan saat ini, seperti batasan area dan posisi musuh.
- Fungsi Obyektif: Maksimalkan pengumpulan diamonds sambil meminimalkan risiko.

2. Analisis Efisiensi Solusi

Strategi ini berfokus pada penilaian dinamis antara eksplorasi area baru untuk menemukan diamonds dan eksploitasi area yang diketahui dengan risiko yang lebih rendah. Dengan menggabungkan pendekatan ini, bot dapat lebih efisien dalam pengambilan keputusan berdasarkan kondisi saat ini. Efisiensi solusi ini tergantung pada seberapa cepat bot dapat menilai kondisi dan membuat keputusan, yang pada umumnya dapat dianggap $O(n)$, dengan n adalah jumlah elemen atau variabel yang perlu dipertimbangkan (misalnya, jumlah diamonds, posisi musuh, dan zona aman).

3. Analisis Efektivitas Solusi

Strategi ini efektif jika:

- Bot dapat secara akurat memprediksi perilaku musuh dan mengidentifikasi area dengan rasio keuntungan risiko yang tinggi.
- Terdapat cukup informasi yang tersedia untuk membuat keputusan berdasarkan penilaian risiko.

Strategi ini kurang efektif jika:

- Peta permainan sangat dinamis dan sulit untuk memprediksi, membuat eksplorasi berisiko tinggi.
- Musuh menggunakan strategi yang sangat agresif atau tidak terduga, yang meningkatkan risiko eksplorasi.

Pendekatan Greedy by Exploration and Exploitation Balance memungkinkan adaptasi terhadap kondisi permainan yang berubah dan memanfaatkan peluang saat muncul, dengan tetap mempertimbangkan risiko yang terlibat. Ini mendorong permainan yang lebih fleksibel dan responsif, dengan mengoptimalkan keuntungan dalam mengumpulkan diamonds sambil menyerang musuh untuk mendapatkan diamonds. Tentunya, dengan pertimbangan berapa diamonds yang musuh bawa saat ini. Akan tidak menguntungkan apabila memprioritaskan untuk menyerang musuh saat bot musuh yang dituju hanya membawa sedikit diamonds.

3.1.2 Greedy by Attacking Enemy Bots

1. Mapping Elemen Greedy
 - Himpunan Kandidat: Semua kemungkinan langkah untuk mengumpulkan diamonds atau menyerang musuh.
 - Himpunan Solusi: Langkah yang mengoptimalkan pengumpulan diamonds sambil meminimalkan risiko terkena serangan.
 - Fungsi Solusi: Memeriksa apakah langkah yang dipilih berhasil menyeimbangkan antara eksplorasi untuk mencari diamonds maksimal ataupun menyerang musuh.
 - Fungsi Seleksi: Memilih langkah yang memberikan rasio terbaik antara potensi keuntungan (mengumpulkan diamonds & menyerang musuh) dan potensi kerugian (terkena serangan).
 - Fungsi Kelayakan: Memeriksa apakah langkah memenuhi kondisi permainan saat ini, seperti batasan area dan posisi musuh.
 - Fungsi Obyektif: Memaksimalkan pengumpulan diamonds serta mengambil risiko untuk menyerang musuh.

2. Analisis Efisiensi Solusi

Strategi ini berfokus pada penilaian dinamis antara eksplorasi area baru untuk mencari posisi musuh terdekat untuk kemudian menyerangnya dengan mengorbankan risiko. Dengan menggabungkan pendekatan ini, bot dapat lebih efisien dalam pengambilan keputusan berdasarkan kondisi saat ini. Efisiensi solusi ini tergantung pada seberapa cepat bot dapat menilai kondisi dan membuat keputusan, yang pada umumnya dapat dianggap $O(n)$, dengan n adalah jumlah elemen atau variabel yang perlu dipertimbangkan.

3. Analisis Efektivitas Solusi

Strategi ini efektif jika:

- Bot dapat secara akurat mengidentifikasi area dan mencari musuh dengan rasio keuntungan risiko yang tinggi.

Strategi ini kurang efektif jika:

- Peta permainan sangat dinamis dan sulit untuk memprediksi, membuat eksplorasi berisiko tinggi.

- Musuh menggunakan strategi yang sangat agresif atau tidak terduga, yang meningkatkan risiko eksplorasi.

3.1.3 Greedy by 3-steps Diamonds

1. Mapping Elemen Greedy
- Himpunan Kandidat: Semua kemungkinan langkah untuk mengumpulkan diamonds dalam 3 langkah.
- Himpunan Solusi: Langkah yang mengoptimalkan pengumpulan diamonds sambil meminimalkan risiko terkena serangan.
- Fungsi Solusi: Memeriksa apakah langkah yang dipilih berhasil menyeimbangkan antara eksplorasi untuk diamonds dan eksploitasi dari posisi saat ini.
- Fungsi Seleksi: Memilih langkah yang memberikan rasio terbaik antara potensi keuntungan (mengumpulkan diamonds) dan potensi kerugian (terkena serangan).
- Fungsi Kelayakan: Memeriksa apakah langkah memenuhi kondisi permainan saat ini, seperti batasan area dan posisi musuh.
- Fungsi Obyektif: Maksimalkan pengumpulan diamonds sambil meminimalkan risiko.

2. Analisis Efisiensi Solusi

Strategi ini berfokus pada penilaian dinamis antara eksplorasi area baru untuk menemukan diamonds dan eksploitasi area yang diketahui dengan risiko yang lebih rendah. Diamonds yang dapat diambil dalam tiga langkah akan diprioritaskan terlebih dahulu dibanding diamonds dengan nilai yang lebih tinggi, namun jaraknya jauh. Dengan menggabungkan pendekatan ini, bot dapat lebih efisien dalam pengambilan keputusan berdasarkan kondisi saat ini. Efisiensi solusi ini tergantung pada seberapa cepat bot dapat menilai kondisi dan membuat keputusan, yang pada umumnya dapat dianggap $O(n)$, dengan n adalah jumlah elemen atau variabel yang perlu dipertimbangkan (misalnya, jumlah diamonds).

3. Analisis Efektivitas Solusi

Strategi ini efektif jika:

- Bot dapat secara akurat memprediksi perilaku musuh dan mengidentifikasi area dengan rasio keuntungan risiko yang tinggi.
 - Terdapat cukup informasi yang tersedia untuk membuat keputusan berdasarkan penilaian risiko.
 - Terdapat cukup banyak diamonds dalam satu cluster (sepetak area yang berisi banyak diamonds).
 - Posisi bot lebih dekat ke cluster yang padat diamonds.
- Strategi ini kurang efektif jika:
- Peta permainan sangat dinamis dan sulit untuk memprediksi, membuat eksplorasi berisiko tinggi.

- Musuh menggunakan strategi yang sangat agresif atau tidak terduga, yang meningkatkan risiko eksplorasi.
- Bot spawn di posisi yang jauh dari cluster padat diamonds.
- Letak diamonds tersebar jauh dan tidak membentuk cluster padat diamonds.
- Posisi musuh lebih dekat menuju ke cluster padat diamonds.

Pendekatan Greedy by 3-steps diamonds memungkinkan adaptasi terhadap kondisi permainan yang berubah dan memanfaatkan peluang saat muncul, dengan tetap mempertimbangkan risiko yang terlibat. Ini mendorong permainan yang lebih fleksibel dan responsif, dengan mengoptimalkan keuntungan dalam mengumpulkan diamonds dan mengutamakan tujuan cluster padat diamonds.

3.2 Strategi Greedy yang Dipilih

3.2.1 Combined Greedy Strategy

1. Mapping Elemen Greedy
- Himpunan Kandidat: Semua kemungkinan langkah untuk mengumpulkan diamonds dalam 3 langkah.
- Himpunan Solusi: Langkah yang mengoptimalkan pengumpulan diamonds sambil meminimalkan risiko terkena serangan.
- Fungsi Solusi: Memeriksa apakah langkah yang dipilih berhasil menyeimbangkan antara eksplorasi untuk diamonds dan eksploitasi dari posisi saat ini.
- Fungsi Seleksi: Memilih langkah yang memberikan rasio terbaik antara potensi keuntungan (mengumpulkan diamonds) dan potensi kerugian (terkena serangan).
- Fungsi Kelayakan: Memeriksa apakah langkah memenuhi kondisi permainan saat ini, seperti batasan area dan posisi musuh.
- Fungsi Obyektif: Maksimalkan pengumpulan diamonds sambil meminimalkan risiko.

2. Analisis Efisiensi Solusi

Strategi ini merupakan gabungan dari strategi konvensional Greedy by Exploration and Exploitation Balance dan Greedy by 3-steps Diamonds. Strategi ini berfokus pada penilaian dinamis antara eksplorasi area baru untuk menemukan diamonds dan eksploitasi area yang diketahui dengan risiko yang lebih rendah. Diamonds yang dapat diambil dalam tiga langkah akan diprioritaskan terlebih dahulu dibanding diamonds dengan nilai yang lebih tinggi, namun jaraknya jauh. Dengan menggabungkan pendekatan ini, bot dapat lebih efisien dalam pengambilan keputusan berdasarkan kondisi saat ini. Efisiensi solusi ini tergantung pada seberapa cepat bot dapat menilai kondisi dan membuat keputusan, yang pada umumnya dapat dianggap $O(n)$, dengan n adalah jumlah elemen atau variabel yang perlu dipertimbangkan (misalnya, jumlah diamonds).

3. Analisis Efektivitas Solusi

Strategi ini efektif jika:

- Bot dapat secara akurat memprediksi perilaku musuh dan mengidentifikasi area dengan rasio keuntungan risiko yang tinggi.
- Terdapat cukup informasi yang tersedia untuk membuat keputusan berdasarkan penilaian risiko.
- Terdapat cukup banyak diamonds dalam satu cluster (sepetak area yang berisi banyak diamonds).
- Posisi bot lebih dekat ke cluster yang padat diamonds.
Strategi ini kurang efektif jika:
 - Peta permainan sangat dinamis dan sulit untuk memprediksi, membuat eksplorasi berisiko tinggi.
 - Musuh menggunakan strategi yang sangat agresif atau tidak terduga, yang meningkatkan risiko eksplorasi.
 - Bot spawn di posisi yang jauh dari cluster padat diamonds.
 - Letak diamonds tersebar jauh dan tidak membentuk cluster padat diamonds.
 - Posisi musuh lebih dekat menuju ke cluster padat diamonds.

Pendekatan Combined Greedy Strategy memungkinkan adaptasi terhadap kondisi permainan yang berubah dan memanfaatkan peluang saat muncul, dengan tetap mempertimbangkan risiko yang terlibat. Dalam strategi combined Greedy ini, tentunya terdapat prioritas antara diamond dengan poin yang lebih tinggi ataupun cluster diamonds dengan juga tetap mempertimbangkan jarak bot terhadap diamond merah ataupun cluster diamonds. Ini mendorong permainan yang lebih fleksibel dan responsif, dengan mengoptimalkan keuntungan dalam mengumpulkan diamonds dan mengutamakan tujuan cluster padat diamonds.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Repository Github

https://github.com/Edvardesu/Tubes1_Tonggoku-Nyaleg-Ndes

4.2 Implementasi dan Pseudocode

Check Blue Function

```
n <- 0  
  
d traversal board.diamonds  
  
if (d.properties.points == 1) then  
  
    n <- n + 1  
  
RETURN n
```

Get Nearest Blue Function

```
IF check_blue(board) > 0 THEN  
  
    in_y, in_x, out_y, out_x, min <- get_portal(board_bot, board)  
  
    temp_min <- 30  
  
    FOR d IN board.diamonds  
  
        IF d.properties.points == 1 THEN  
  
            delta <- abs(d.position.y - out_y) + abs(d.position.x - out_x) + abs(in_y -  
            board_bot.position.y) + abs(in_x - board_bot.position.x)  
  
            IF delta < temp_min THEN  
  
                temp_min <- delta  
  
            min <- min + temp_min  
  
            res_y <- in_y  
  
            res_x <- in_x  
  
        FOR d IN board.diamonds  
  
            IF d.properties.points == 1 THEN  
  
                delta <- abs(d.position.y - board_bot.position.y) + abs(d.position.x -  
                board_bot.position.x)  
  
                IF delta < min THEN  
  
                    min <- delta  
  
                    res_y <- d.position.y  
  
                    res_x <- d.position.x  
  
                IF res_y == board_bot.position.y AND res_x == board_bot.position.x THEN  
  
                    res_y <- res_y + 1  
  
    RETURN res_y, res_x, min
```

Check Red Function

```
n <- 0  
FOR d IN board.diamonds  
    IF d.properties.points == 2 THEN  
        n <- n + 1  
RETURN n
```

Get Nearest Red Function

```
IF check_red(board) > 0 THEN  
    in_y, in_x, out_y, out_x, min <- get_portal(board_bot, board)  
    temp_min <- 30  
    TRAVERSE d IN board.diamonds  
        IF d.properties.points == 2 THEN  
            delta <- ABS(d.position.y - out_y) + ABS(d.position.x - out_x)  
            IF delta < temp_min THEN  
                temp_min <- delta  
            min <- min + temp_min  
            res_y <- in_y  
            res_x <- in_x  
        TRAVERSE d IN board.diamonds  
            IF d.properties.points == 2 THEN  
                delta <- ABS(d.position.y - board_bot.position.y) + ABS(d.position.x -  
                board_bot.position.x)  
                IF delta < min THEN  
                    min <- delta  
                    res_y <- d.position.y  
                    res_x <- d.position.x  
            IF res_y == board_bot.position.y AND res_x == board_bot.position.x THEN  
                res_y <- res_y + 1  
RETURN res_y, res_x, min
```

Get Portal Function

```
min <- 30  
  
in_y <- -1  
  
in_x <- -1  
  
out_y <- -1  
  
out_x <- -1  
  
delta0 <- ABS(board.game_objects[0].position.y - board_bot.position.y) +  
ABS(board.game_objects[0].position.x - board_bot.position.x)  
  
delta1 <- ABS(board.game_objects[1].position.y - board_bot.position.y) +  
ABS(board.game_objects[1].position.x - board_bot.position.x)  
  
IF delta0 < delta1 THEN  
    in_y <- board.game_objects[0].position.y  
    in_x <- board.game_objects[0].position.x  
    out_y <- board.game_objects[1].position.y  
    out_x <- board.game_objects[1].position.x  
    min <- delta0  
  
ELSE  
    in_y <- board.game_objects[1].position.y  
    in_x <- board.game_objects[1].position.x  
    out_y <- board.game_objects[0].position.y  
    out_x <- board.game_objects[0].position.x  
    min <- delta1  
  
RETURN in_y, in_x, out_y, out_x, min
```

Go Home Function

```
base <- board_bot.properties.base  
  
delta <- ABS(board_bot.position.y - base.y) + ABS(board_bot.position.x - base.x)  
  
RETURN delta
```

Go Button Function

```
{Menemukan posisi tombol berlian}

y_button <- -1
x_button <- -1

g traversal board.game_objects

if (g.type == "DiamondButtonGameObject") then
    y_button <- g.position.y
    x_button <- g.position.x

RETURN y_button, x_button
```

Threesteps Function

```
min <- 4 {tidak ada dalam 3 langkah dari posisi bot}
xtemp <- 0 {untuk tampung nilai dengan jarak terkecil}
ytemp <- 0 {untuk tampung nilai dengan jarak terkecil}
base <- board_bot.properties.base

d traversal board.diamonds
if (d.properties.points == 1 or d.properties.points == 2) then
    jarakx <- abs(d.position.x - board_bot.position.x)
    jaraky <- abs(d.position.y - board_bot.position.y)
    if (jarakx + jaraky < min) {dalam tiga langkah}
        xtemp <- d.position.x
        ytemp <- d.position.y
    elseif (jarakx + jaraky == min) {sama dengan minimum}
        tempbasedistance <- abs(xtemp-base.x) + abs(ytemp-base.y)
        currentbasedistance <- abs(d.position.x-base.x) + abs(d.position.y-base.y)
        if (currentbasedistance <= tempbasedistance)
            xtemp <- d.position.x
            ytemp <- d.position.y
    elseif (jarakx + jaraky == 1) {hanya beda 1 kotak sudah pasti minimum}
        RETURN ytemp, xtemp

RETURN ytemp, xtemp
```

Check Portal And Base Align Function

```
check <- False
base_x <- board_bot.properties.base.x
base_y <- board_bot.properties.base.y
portal_y0 <- board.game_objects[0].position.y
portal_y1 <- board.game_objects[1].position.y
portal_x0 <- board.game_objects[0].position.x
portal_x1 <- board.game_objects[1].position.x

if (board_bot.goal_position.y == base_y) then
    if (board_bot.position.y == base_y and board_bot.position.y == portal_y0) then
        if (board_bot.position.y < portal_y0 < base_y) then
            check <- True
        elseif (board_bot.position.y > portal_y1 > base_y) then
            check <- True
        elseif (board_bot.position.y == base_y and board_bot.position.y == portal_y1) then
            if (board_bot.position.y < portal_y0 < base_y) then
                check <- True
            elseif (board_bot.position.y > portal_y1 > base_y) then
                check <- True

    if (board_bot.goal_position.x == base_x) then
        if (board_bot.position.x == base_x and board_bot.position.x == portal_x0) then
            if (board_bot.position.x < portal_x0 < base_x) then
                check <- True
            elseif (board_bot.position.x > portal_x1 > base_x) then
                check <- True
            elseif (board_bot.position.x == base_x and board_bot.position.x == portal_x1) then
                if (board_bot.position.x < portal_x0 < base_x) then
                    check <- True
                elseif (board_bot.position.x > portal_x1 > base_x) then
                    check <- True

RETURN check
```

Evade Portal Function

```

res_y <- board_bot.position.y
res_x <- board_bot.position.x

if (check_portal_and_base_align(board_bot, board)) then
    if (board_bot.position.y == 14) then
        res_y <- res_y - 1
    elseif (board_bot.position.y == 0) then
        res_y <- res_y + 1
    elseif (board_bot.position.x == 14) then
        res_x <- res_x - 1
    elseif (board_bot.position.x == 0) then
        res_x <- res_x + 1

RETURN res_y, res_x

```

4.3 Struktur Data Program

Bahasa yang digunakan dalam pembuatan bot adalah bahasa pemrograman Python. Struktur data pada program didefinisikan dengan menggunakan class. Class yang digunakan dalam pengembangan bot terdapat pada folder Logic pada file mybot.py.

```

128 < class MyBot(BaseLogic):
129     Codeium: Refactor | Explain | Generate Docstring | X
130     def __init__(self):
131         # Initialize attributes necessary
132         self.directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
133         self.goal_position: Optional[Position] = None
134         self.current_direction = 0
135     Codeium: Refactor | Explain | Generate Docstring | X
136     def next_move(self, board_bot: GameObject, board: Board):
137         props = board_bot.properties
138         time = math.floor(board_bot.properties.milliseconds_left / 1000)
139         y_button, x_button = go_button(board)
140         if check_blue(board) + check_red(board) < 17 and (abs(board_bot.position.y - y_button) + abs(board_bot.position.x - x_button)) < 5 and props.diamonds < 4:
141             y, x = go_button(board)
142             target = [y, x]
143             self.goal_position = target
144         else:
145             if props.diamonds == 5:
146                 base = board_bot.properties.base
147                 target = [base.y, base.x]
148                 self.goal_position = target
149             else:
150                 if props.diamonds == 3 or props.diamonds == 2:
151                     if go_home(board_bot) == time - 3:
152                         base = board_bot.properties.base
153                         target = [base.y, base.x]
154                         self.goal_position = target
155                     elif go_home(board_bot) > time - 5:
156                         res_y, res_x, min, time, carry = get_nearest_enemy(board_bot, board)
157                         if carry > 0 and min == 1 and board_bot.properties.milliseconds_left - time > 0:
158                             target = [int(res_y), int(res_x)]
159                             self.goal_position = target
160                         else:
161                             if check_red(board) > 0 and check_blue(board) > 0:

```

Class yang terdapat pada mybot.py hanya satu, yaitu class MyBot yang mengextend BaseLogic memiliki fungsi utama untuk menentukan gerakan selanjutnya dari bot tersebut berdasarkan kondisi saat ini di papan permainan. Garis besar dari myBot adalah sebagai berikut:

1. Inisialisasi (init): Bot diinisialisasi dengan arah gerakan yang mungkin, posisi tujuan (goal_position) yang belum ditentukan, dan arah saat ini. Arah gerakan didefinisikan sebagai pasangan koordinat (delta_x, delta_y) untuk empat arah: kanan, bawah, kiri, atas.
2. Menentukan Langkah Selanjutnya (next_move):
 - Bot memeriksa properti dan kondisi saat ini, termasuk sisa waktu, jumlah berlian yang dimiliki, dan posisi tombol.
 - Bot memiliki beberapa kondisi yang mempengaruhi pilihan target atau tujuan selanjutnya:
 - Jika jumlah berlian yang belum terkumpul kurang dari 17 dan bot dekat dengan tombol sambil memiliki kurang dari 4 berlian, bot akan menuju tombol.
 - Jika bot memiliki 5 berlian, bot akan kembali ke basisnya.
 - Dalam kondisi tertentu berdasarkan waktu tersisa dan jumlah berlian, bot akan memutuskan untuk kembali ke basis, mengejar musuh terdekat, atau mengumpulkan berlian merah atau biru terdekat.
 - Logika untuk memilih antara berlian merah atau biru tergantung pada jarak dan kondisi lainnya, seperti jumlah total berlian merah dan biru yang tersisa.
 - Selain itu, ada kondisi yang memerintahkan bot untuk "berkelana" jika tidak ada tujuan spesifik yang ditetapkan, dengan memilih arah secara acak dari empat arah yang mungkin.
3. Kalkulasi Delta untuk Pergerakan:
 - Jika goal_position ditentukan, bot akan menghitung perbedaan (delta) antara posisi saat ini dan tujuan untuk menentukan arah pergerakan.
 - Jika tidak, bot akan bergerak berdasarkan arah saat ini dan akan mengubah arahnya secara acak.

Secara keseluruhan, logika bot ini dirancang untuk membuat keputusan berdasarkan kondisi saat ini dari papan permainan, posisinya, sisa waktu, dan jumlah berlian yang dimiliki. Bot mencoba mengoptimalkan tindakannya untuk mengumpulkan berlian, menghindari atau mengejar musuh, dan kembali ke basisnya dengan mempertimbangkan kondisi dinamis dari permainan.

4.4 Analisis

Kami melakukan analisis dan pengujian terhadap myBot (atau disebut dengan Bot pada analisis) dengan cara mempertarungkan myBot dengan bot lawan dari bot dengan algoritma alternatif lain ataupun random bot yang sudah disediakan oleh asisten. Pada contoh pengujian yang dilampirkan pada bawah ini, myBot melawan 3 buah bot yaitu dari bot stima, stima1, stima 2, dan stima 3. Dilakukan pengujian terhadap beberapa aksi yang dapat dilakukan oleh bot. Berbagai skenario dengan mempertimbangkan tiap object adalah sebagai berikut.

4.4.1 Blue Diamonds ('get_nearest_blue')

Bot mencari dan mendekati blue diamond terdekat. Ini dilakukan dengan membandingkan jarak dari bot ke setiap blue diamond, termasuk mempertimbangkan

penggunaan teleporter jika itu akan mempersingkat jarak.

4.5 Red Diamonds ('get_nearest_red')

Strategi serupa dengan blue diamonds, tapi untuk red diamonds bot akan memprioritaskan mendekati dan mengambil red diamond terdekat berdasarkan perhitungan jarak yang serupa, termasuk mempertimbangkan teleporter.

4.4.3 Teleporter ('get_portal')

Bot menggunakan teleporter untuk mempersingkat jarak ke targetnya, baik itu diamond atau objek lain. Fungsi get_portal menghitung portal terdekat dan jarak ke portal tersebut, serta output portal untuk memperkirakan penggunaan portal yang paling efisien.

4.4.4 Red Button ('go_button')

Bot mendekati dan mengaktifkan red button jika jumlah total blue dan red diamonds di peta kurang dari 17 dan bot berada dalam jarak kurang dari 5 unit dari tombol, serta bot memiliki kurang dari 4 diamonds. Ini kemungkinan bertujuan untuk mengaktifkan mekanisme tertentu dalam permainan (misalnya, mengeluarkan lebih banyak diamonds).

4.4.5 Bases ('go_home')

Strategi "pulang" diaktifkan ketika bot memiliki jumlah diamonds tertentu (2, 3, atau 5) dan waktu yang tersisa cukup dekat dengan waktu yang dibutuhkan untuk kembali ke base. Ini untuk memastikan bot bisa kembali ke base untuk mengamankan diamonds yang telah dikumpulkan.

4.4.6 Enemy Bot ('get_nearest_enemy')

Bot mengidentifikasi dan mendekati musuh terdekat jika kondisi tertentu terpenuhi, seperti jika musuh membawa diamond dan bot bisa mengalahkannya dalam waktu. Strategi ini terutama diaktifkan ketika bot sedang tidak dalam misi mengumpulkan diamond atau dalam kondisi tertentu berdasarkan waktu yang tersisa dan jarak ke base sendiri.

4.4.7 Evade Portal ('evade_portal')

Fungsi evade_portal dirancang untuk menghindari portal ketika posisi portal dan base (basis) dari bot berada dalam suatu garis lurus yang sama, baik secara horizontal maupun vertikal. Fungsi ini sangat penting dalam strategi permainan untuk menghindari penggunaan portal yang tidak perlu atau yang bisa mengganggu strategi utama, terutama ketika bot ingin kembali ke base dengan cepat tanpa terlempar ke tempat lain oleh portal. Jika kondisi di atas terpenuhi, maka bot akan mencoba menghindar dengan cara memindahkan posisinya satu langkah ke arah yang aman. Cara ini bertujuan untuk memastikan bot tidak secara tidak sengaja memasuki portal saat dalam perjalanan kembali ke base.

- Logika penghindaran mempertimbangkan batas-batas papan permainan. Misalnya, jika

bot berada di tepi paling atas papan ($y = 0$), maka bot akan bergerak ke bawah (menambah y), dan sebaliknya.

- Logika serupa diterapkan untuk kasus di tepi kiri, kanan, dan bawah papan permainan.

4.4.2 Pengujian

Pada tahap pengujian akhir, dilakukan tes permainan terhadap 4 bot yang dijalankan sekaligus dengan dua bot memakai algoritma random yang tersedia secara default pada bot starter pack dan dua bot lainnya menggunakan combined greedy algorithm yang telah dibuat oleh kelompok kami. Pengujian dilakukan dalam rangka mengetahui apakah algoritma bot sudah cukup efisien dan memiliki strategi yang bagus serta dapat menghasilkan diamonds maksimal demi memenangkan permainan.

```
C:\Windows\system32\cmd.e: <-- 200 OK
54
>>> POST /bots/7bc8f46e-252b-4cca-af41-b5940621b9de/move {'direction': 'EAST'}
|
<<< 200 OK
53
>>> POST /bots/7bc8f46e-252b-4cca-af41-b5940621b9de/move {'direction': 'WEST'}
|
<<< 200 OK
52
>>> POST /bots/7bc8f46e-252b-4cca-af41-b5940621b9de/move {'direction': 'WEST'}
|
<<< 200 OK
51
>>> POST /bots/7bc8f46e-252b-4cca-af41-b5940621b9de/move {'direction': 'WEST'}
|
<<< 200 OK

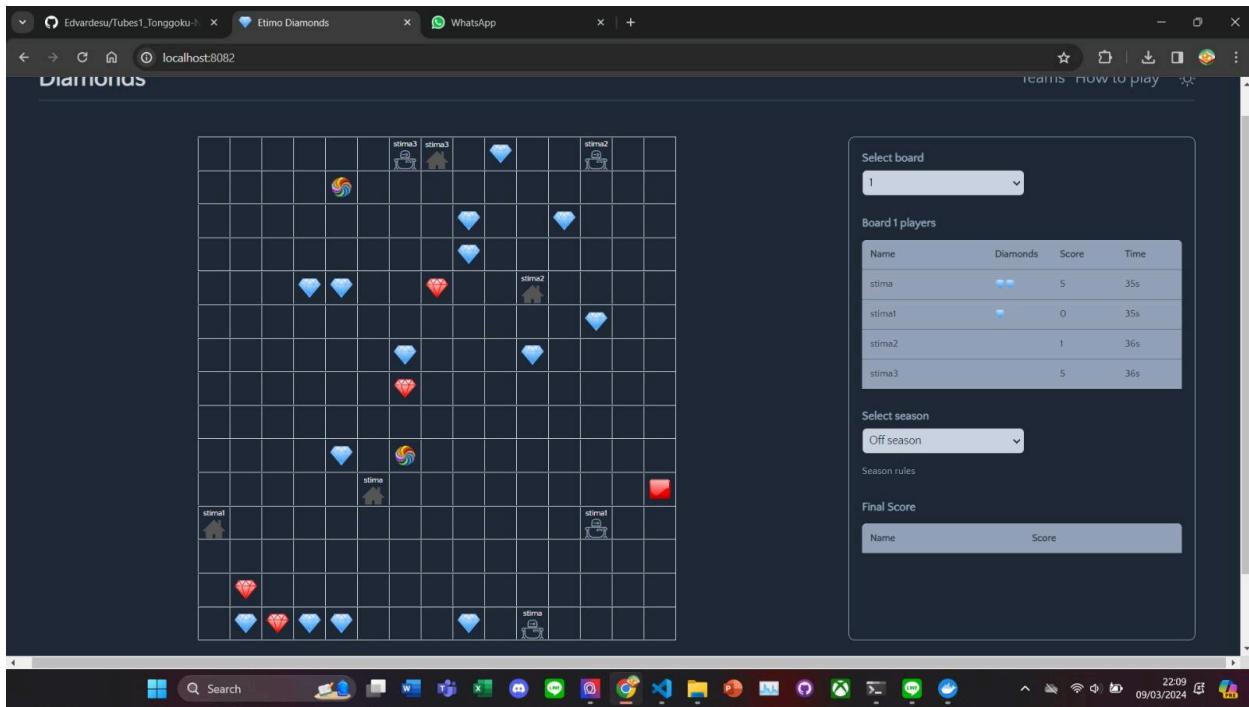
C:\Windows\system32\cmd.e: <-- 200 OK
100
>>> POST /bots/648c99d0-33e0-4d55-b147-746f18de71d6/move {'direction': 'SOUT
H'}
|
<<< 200 OK
100
>>> POST /bots/648c99d0-33e0-4d55-b147-746f18de71d6/move {'direction': 'SOUT
H'}
|
<<< 200 OK
100
>>> POST /bots/648c99d0-33e0-4d55-b147-746f18de71d6/move {'direction': 'SOUT
H'}
|
<<< 200 OK
100
>>> POST /bots/648c99d0-33e0-4d55-b147-746f18de71d6/move {'direction': 'SOUT
H'}
|
<<< 200 OK

C:\Windows\system32\cmd.e: <-- 200 OK
100
>>> POST /bots/7f3e80d4-b79f-49b5-948b-86359847bd97/move {'direction': 'WEST
H'}
|
<<< 200 OK
100
>>> POST /bots/7f3e80d4-b79f-49b5-948b-86359847bd97/move {'direction': 'NORT
H'}
|
<<< 200 OK
100
>>> POST /bots/7f3e80d4-b79f-49b5-948b-86359847bd97/move {'direction': 'EAST
H'}
|
<<< 200 OK
100
>>> POST /bots/7f3e80d4-b79f-49b5-948b-86359847bd97/move {'direction': 'SOUT
H'}
|
<<< 200 OK

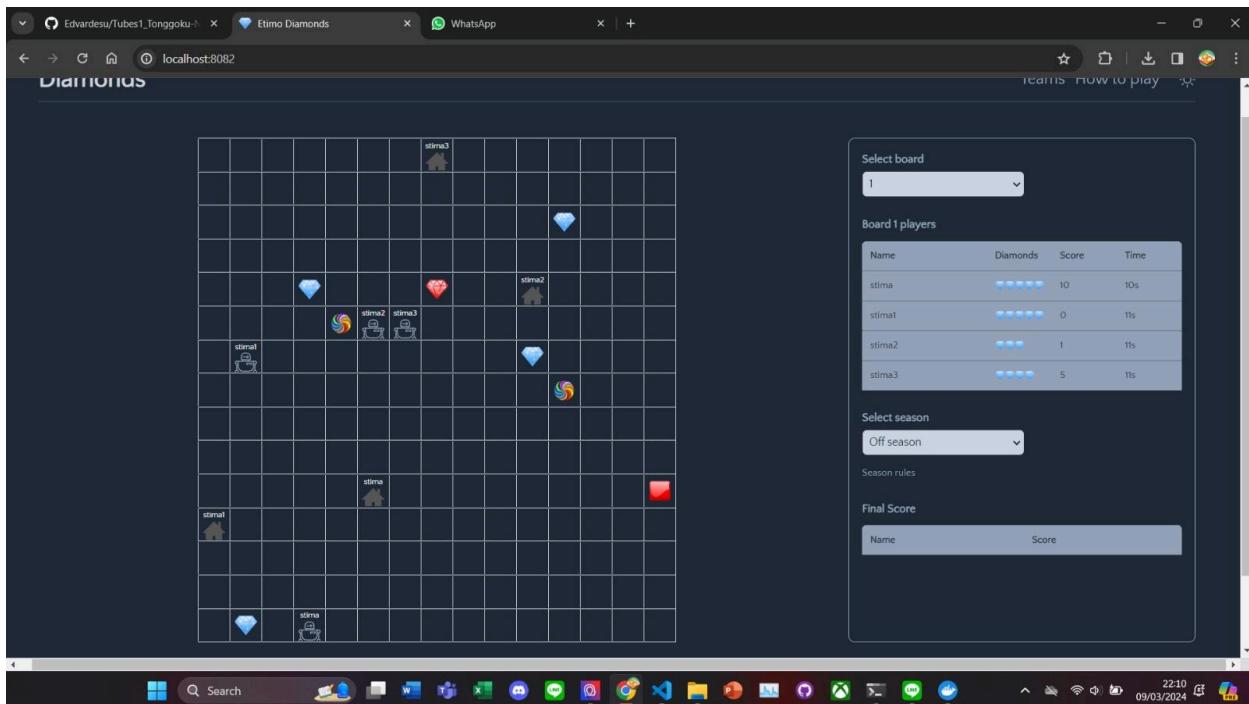
C:\Windows\system32\cmd.e: <-- 200 OK
100
>>> POST /bots/507412db-352a-41a1-bc97-279632ae42c7/move {'direction': 'NORT
H'}
|
<<< 200 OK
53
>>> POST /bots/507412db-352a-41a1-bc97-279632ae42c7/move {'direction': 'NORT
H'}
|
<<< 200 OK
52
>>> POST /bots/507412db-352a-41a1-bc97-279632ae42c7/move {'direction': 'EAST
H'}
|
<<< 200 OK
51
>>> POST /bots/507412db-352a-41a1-bc97-279632ae42c7/move {'direction': 'SOUT
H'}
|
<<< 200 OK
```

Permainan dimulai dengan layout board dan tiap objects sebagai berikut. Bot stima dan stima1 adalah bot buatan kami. Bot stima2 dan stima3 adalah bot random bawaan dari bot starter pack. Pada screenshot di bawah ini dapat dilihat bahwa pada awal permainan, bot stima dan stima1 sudah menghasilkan lebih banyak diamonds.

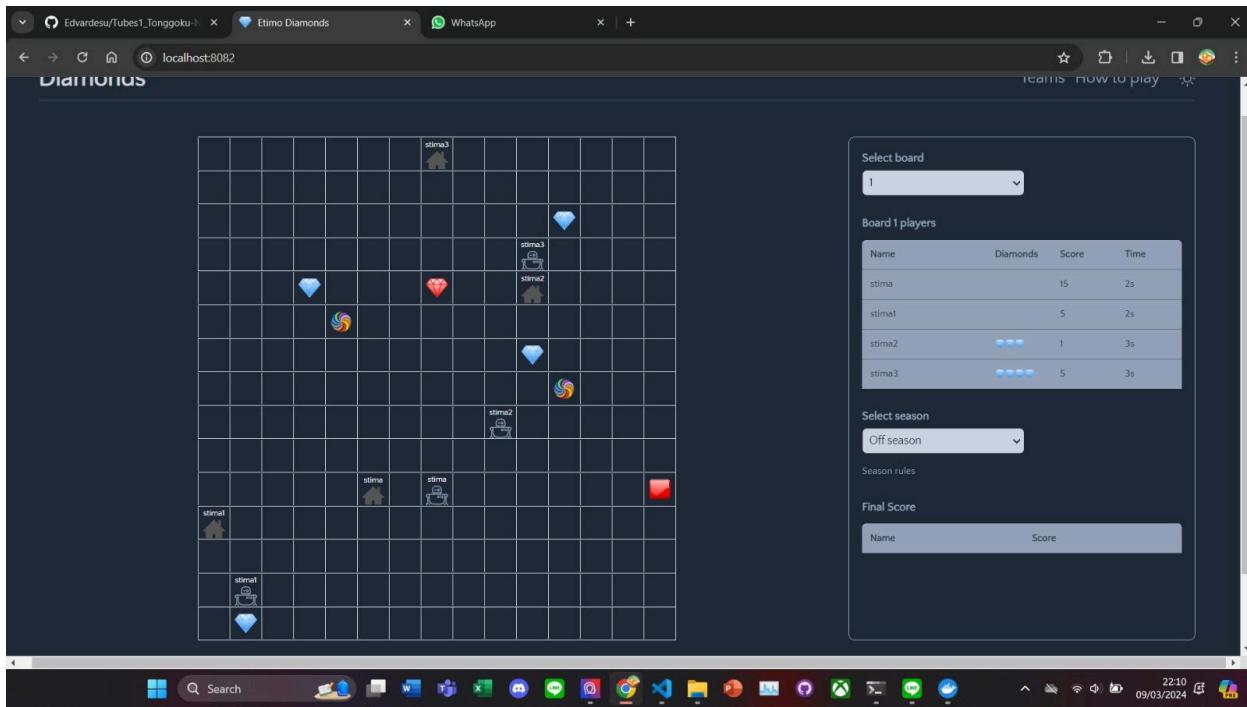
IF2211 Strategi Algoritma – Tugas Besar 1



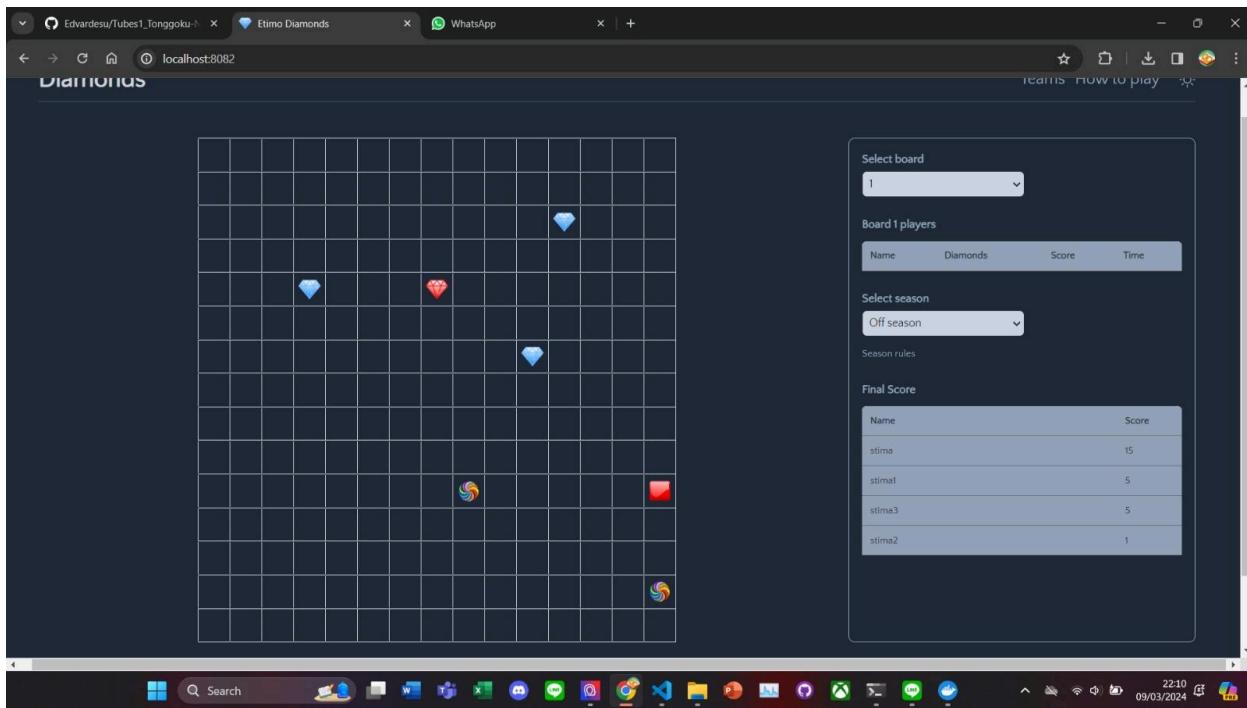
Pada gambar dibawah terbukti bahwa bot stima dan stima1 sudah mengumpulkan lebih banyak diamonds dibanding bot stima2 dan stima3.



IF2211 Strategi Algoritma – Tugas Besar 1



Pada gambar dibawah adalah screenshot dari hasil akhir permainan. Didapat bahwa bot yang terbanyak diamondsnya adalah bot stima (bot yang kami buat). Hal ini membuktikan bahwa algoritma bot yang dibuat sudah cukup bagus dengan memperoleh hasil yang memuaskan. Algoritma bot sudah cukup efisien dan strategi yang diimplementasikan sudah cukup efektif dalam memenangkan pertandingan.



BAB V

KESIMPULAN DAN SARAN

Dari tugas besar IF2211 Strategi Algoritma, telah berhasil diciptakan sebuah bot permainan Diamonds menggunakan strategi algoritma Greedy. Strategi yang digunakan tidak hanya satu, melainkan kombinasi dari beberapa strategi yang memiliki prioritasnya masing-masing sehingga terciptalah sebuah bot dengan strategi terbaik Combined Greedy Strategy.

Algoritma greedy terbukti digunakan untuk membuat bot yang cukup decent untuk permainan Diamonds karena solusi yang dipilih merupakan optimum lokal yang nantinya digunakan untuk mendapat optimum global. Implementasi algoritma greedy pada MyBot telah berhasil membuat bot ini memenangkan match melawan bot-bot lain seperti bot random secara konsisten. Bot ini berhasil menghindari portal yang berada di satu garis dan mendapatkan diamonds dengan mempertimbangkan 3-steps serta jarak menuju diamond terdekat.

Namun, terkadang algoritma greedy gagal menemukan solusi optimum global yang mengakibatkan bot gagal mendapatkan keuntungan terbesar. Beberapa match menunjukkan bot gagal karena mendapatkan lokasi yang kurang menguntungkan sedangkan bot lawan lebih strategis posisinya. Akan, tetapi secara garis besar kelompok kami telah berhasil menciptakan bot yang memiliki algoritma yang efisien dan efektif untuk memenangkan pertandingan.

Saran pengembangan untuk tugas besar ini adalah:

1. Lebih komunikatif agar ide-ide yang dimiliki dapat dipahami oleh seluruh anggota tim dan terealisasikan.
2. Lebih proaktif untuk menghubungi sesama anggota kelompok dan melengkapi bagian yang kurang.
3. Lebih memikirkan alternatif strategi yang digunakan agar dapat lebih di optimasi baik dari segi format kode sampai efektivitas kode.
4. Lebih melakukan manajemen waktu dengan lebih baik lagi agar penggerjaan video dan laporan tidak mendekati hari-H deadline.
5. Lebih menggunakan Github untuk update kode program karena selama ini update kode hanya dilakukan lewat chat Line.

DAFTAR PUSTAKA

<https://informatika.stei.itb.ac.id/~rinaldi.munir/>

REPOSITORY

https://github.com/Edvardesu/Tubes1_Tonggoku-Nyaleg-Ndes

YOUTUBE VIDEO

<https://meet.google.com/zgm-jtur-ngf>