Laporan Tugas Besar 1
IF2211 Strategi Algoritma
Etimo Diamond
Semester II Tahun 2023/2024

Disusun Oleh:
Muhammad Althariq Fairuz 13522027
Farhan Nafis Rayhan 13522037
Randy Verdian 13522067

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

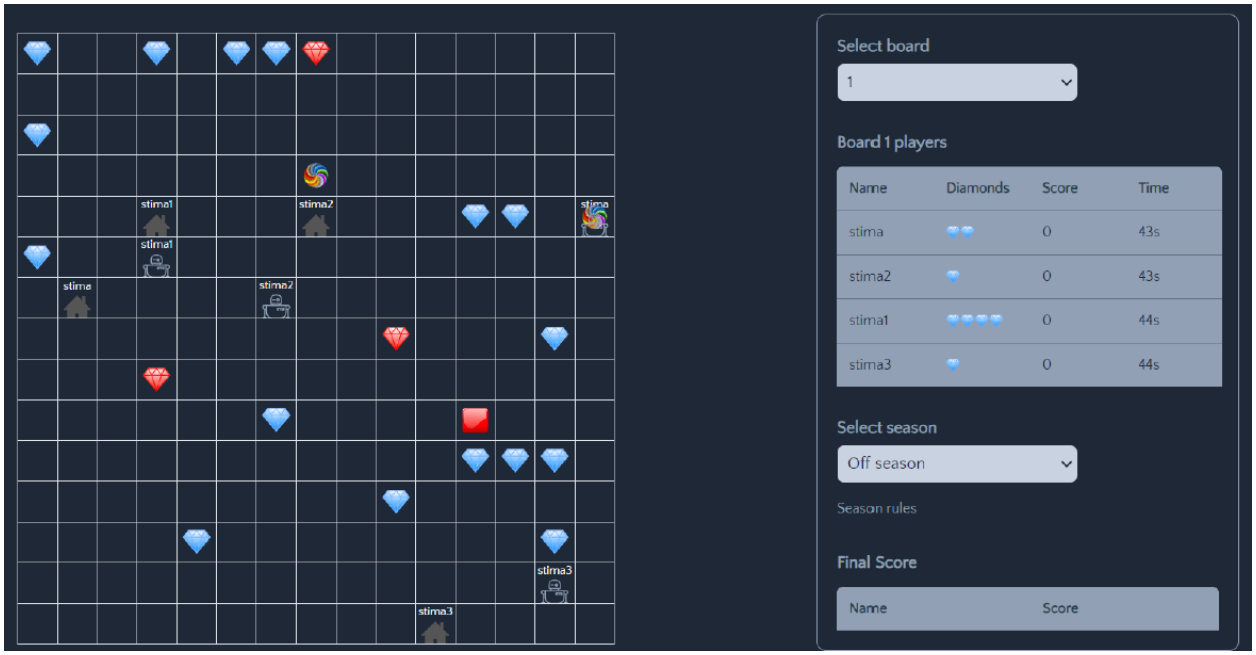
2024

DAFTAR ISI

BAB 1

Deskripsi Tugas

Etimo 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.



Gambar 1.1 Tampilan permainan Etimo Diamond

Program permainan Etimo Diamonds terdiri atas:

1. Game Engine, yang secara umum berisi:
    a. Kodi Backend permainan, yang berisi logic permainan secara keseluruhan serta API yang disediakan untuk berkomunikasi dengan frontend dan program bot
    b. Kodi 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

Untuk mengetahui flow dari game ini, berikut ini adalah cara kerja permainan Etimo Diamonds:

1. Pertama, setiap pemain (bot) akan ditempatkan pada board secara random. Masing-masing bot akan mempunyai home base, serta memiliki score dan inventory awal bernilai nol.
2. Setiap bot diberikan waktu untuk bergerak, waktu yang diberikan semua sama untuk setiap pemain.
3. Objektif utama bot adalah mengambil diamond-diamond yang ada di peta sebanyak-banyaknya. Seperti yang sudah disebutkan di atas, diamond yang berwarna merah memiliki 2 poin dan diamond yang berwarna biru memiliki 1 poin.
4. Setiap bot juga memiliki sebuah inventory, dimana inventory berfungsi sebagai tempat penyimpanan sementara diamond yang telah diambil. Inventory ini sewaktu-waktu bisa penuh, maka dari itu bot harus segera kembali ke home base.
5. Apabila bot menuju ke posisi home base, score bot akan bertambah senilai diamond yang tersimpan pada inventory dan inventory bot akan menjadi kosong kembali.
6. Usahakan agar bot anda tidak bertemu dengan bot lawan. Jika bot A menimpa posisi bot B, bot B akan dikirim ke home base dan semua diamond pada inventory bot B akan hilang, diambil masuk ke inventory bot A (istilahnya tackle).
7. Selain itu, terdapat beberapa fitur tambahan seperti teleporter dan red button yang dapat digunakan apabila anda menuju posisi objek tersebut.
8. Apabila waktu seluruh bot telah berakhir, maka permainan berakhir. Score masing-masing pemain akan ditampilkan pada tabel Final Score di sisi kanan layar.

BAB 2
Landasan Teori

2.1. Dasar Teori

Algoritma greedy, juga dikenal sebagai algoritma rakus, adalah algoritma yang memecahkan persoalan secara langkah per langkah sedemikian sehingga pada setiap langkah diharapkan 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.

2.2. Alur Kerja Program

Program dimulai dengan bot bergabung pertama kali ke dalam permainan, bot juga menerima informasi tentang papan permainan, termasuk posisi diamonds dan bot lawan. Bot bergerak berdasarkan strategi yang sudah diberikan oleh tiap pemain sebelumnya, setiap pemain dapat memberikan strategi kepada bot mereka dengan menambahkan kode pada file random.py pada fungsi *next_move* atau bisa menambahkan file baru pada folder *logic.* File yang berisi strategi untuk bot harus dimasukkan ke dalam *CONTROLLERS* pada file main.py. Setelah permainan dimulai, bot akan bergerak sesuai dengan strategi yang telah diimplementasikan dalam kurun waktu tertentu. Bot yang memiliki skor tertinggi akan memenangkan permainan ini.

BAB 3
Aplikasi Strategi *Greedy*

3.1. Mapping Persoalan Diamonds

Collecting Diamonds yang terdekat dengan Bot

1. Himpunan Kandidat : Himpunan semua diamonds yang tersedia di *board*
2. Himpunan Solusi : Diamond yang telah didapat oleh bot
3. Fungsi Solusi : Jika jumlah diamonds sudah 5, maka bot balik ke *base* dan mengumpulkan poin
4. Fungsi Seleksi : Memilih posisi diamonds yang lebih dekat dengan bot
5. Fungsi Kelayakan : Memeriksa apakah pada sebuah kotak terdapat diamond atau tidak
6. Fungsi Objektif : Minimalisasi total jarak yang ditempuh untuk mencapai 5 diamonds

Collecting Diamonds yang terdekat dengan Base

1. Himpunan Kandidat : Himpunan semua diamonds yang tersedia di *board*
2. Himpunan Solusi : Diamond yang telah didapat oleh bot
3. Fungsi Solusi : Jika jumlah diamonds sudah 5, maka bot balik ke *base* dan mengumpulkan poin
4. Fungsi Seleksi : Memilih diamonds yang lebih dekat dengan base untuk setiap *move* yang dilakukan oleh bot
5. Fungsi Kelayakan : Memeriksa apakah pada sebuah kotak terdapat diamond atau tidak
6. Fungsi Objektif : Minimalisasi total jarak yang ditempuh untuk mencapai 5 diamonds agar waktu yang dibutuhkan untuk balik ke base lebih cepat

Chasing Enemy

1. Himpunan Kandidat : Himpunan posisi musuh pada *board*
2. Himpunan Solusi : Himpunan posisi musuh terdekat yang memiliki diamonds untuk di-*tackle*
3. Fungsi Solusi : Men-*tackle* bot lawan dan memperoleh diamonds yang dimiliki oleh nya
4. Fungsi Seleksi : Memilih bot lawan yang terdekat dan memilih bot yang memiliki jumlah diamond yang jika dikali 5 tidak kurang dari jarak diamond terdekat dari bot

5. Fungsi Kelayakan : Memeriksa apakah setiap kotak pada *board* terdapat bot musuh atau tidak
6. Fungsi Objektif : Men-*tackle* bot musuh dan memperoleh diamonds miliknya

Attack Enemy

1. Himpunan Kandidat : Himpunan posisi musuh pada *board*
2. Himpunan Solusi : Himpunan posisi musuh terdekat yang selisihnya satu satuan dengan bot
3. Fungsi Solusi : Men-*tackle* bot lawan dan memperoleh diamonds yang dimiliki oleh nya
4. Fungsi Seleksi : Memilih bot lawan yang selisihnya satuan baik horizontal maupun vertikal dengan bot kita
5. Fungsi Kelayakan : Memeriksa apakah setiap kotak pada *board* terdapat bot musuh atau tidak
6. Fungsi Objektif : Men-*tackle* bot musuh dan memperoleh diamonds miliknya

Defend

1. Himpunan Kandidat : Himpunan posisi musuh pada *board*
2. Himpunan Solusi : Posisi tujuan untuk menghindar dari serangan bot musuh
3. Fungsi Solusi : Menentukan posisi tujuan dengan menggeser satu satuan ke arah yang berlawanan dengan kemungkinan pergerakan dari bot musuh
4. Fungsi Seleksi : Memilih bot lawan yang selisihnya satuan baik horizontal maupun vertikal dengan bot kita
5. Fungsi Kelayakan : Memeriksa apakah setiap kotak pada *board* terdapat bot musuh atau tidak
6. Fungsi Objektif : Menghindari dari kemungkinan serangan bot musuh

Using Teleport

1. Himpunan Kandidat : Himpunan posisi *teleport*
2. Himpunan Solusi : Posisi teleport yang dipilih sebagai *goal position* dari bot
3. Fungsi Solusi : Menentukan apakah dapat menggunakan teleport saat balik ke base dengan membandingkan jarak jika menggunakan teleport atau tanpa teleport
4. Fungsi Seleksi : Memilih teleport yang terdekat dengan bot
5. Fungsi Kelayakan : Mengecek apakah setiap kotak pada *board* terdapat teleport atau tidak
6. Fungsi Objektif : Minimalisasi total jarak yang ditempuh untuk balik ke *base* menggunakan teleport

Using Red Button

1. Himpunan Kandidat : Himpunan posisi *red button* pada *board*
2. Himpunan Solusi : Posisi *red button* yang dipilih sebagai *goal position* dari bot
3. Fungsi Solusi : Menentukan apakah menggunakan *red button* atau tidak dengan mengecek apakah jarak diamond kebanyakan jauh dari bot
4. Fungsi Seleksi : Memilih *red button* yang terdekat dengan bot
5. Fungsi Kelayakan : Memeriksa apakah jarak diamond terdekat dengan base lebih dari 5 dan jarak *red button* dengan *base* kurang dari jarak diamond terdekat dengan base untuk menentukan apakah dapat menggunakan *red button* atau tidak
6. Fungsi Objektif : Menggunakan *red button* untuk mengubah distribusi diamond pada board

Collect & Chase

1. Himpunan Kandidat : Himpunan posisi seluruh diamond dan musuh pada *board.*
2. Himpunan Solusi :Himpunan posisi musuh terdekat yang memiliki diamonds untuk di-*tackle* dan Himpunan diamond terdekat dari bot.
3. Fungsi Solusi : Jika jumlah diamonds sudah 5, maka bot balik ke *base* dan mengumpulkan poin atau men-tackle lawan untuk mencuri diamond mereka.
4. Fungsi Seleksi : Memilih antara diamond terdekat atau musuh terdekat.
5. Fungsi Kelayakan : Memeriksa apakah pada sebuah kotak terdapat diamond atau musuh ataupun tidak.
6. Fungsi Objektif : Minimalisasi total jarak yang ditempuh untuk mencapai 5 diamonds atau men-tackle bot musuh.

Full Tackle

1. Himpunan Kandidat : Himpunan posisi seluruh musuh pada *board.*
2. Himpunan Solusi :Himpunan posisi musuh terdekat yang memiliki diamonds untuk di-*tackle*.
3. Fungsi Solusi : Men-tackle lawan untuk mencuri diamond mereka atau menggeser satu satuan ke arah yang berlawanan dengan kemungkinan pergerakan dari bot musuh.
4. Fungsi Seleksi : Memilih musuh terdekat yang jaraknya berbeda 1 kotak.
5. Fungsi Kelayakan : Memeriksa apakah musuh yang terdekat memiliki diamond lebih banyak dibandingkan kita atau tidak.
6. Fungsi Objektif : Men-tackle bot musuh atau Menghindari dari kemungkinan serangan bot musuh.

Begal

1. Himpunan Kandidat : Himpunan seluruh musuh dan base mereka.
2. Himpunan Solusi :Himpunan posisi base musuh dan banyaknya diamond setiap musuh.
3. Fungsi Solusi : Men-tackle musuh di depan base mereka saat mereka akan kembali ke base.
4. Fungsi Seleksi : Memilih musuh yang sedang membawa diamond terbanyak.
5. Fungsi Kelayakan : Memeriksa apakah musuh tersebut sudah dekat dengan base (pulang).
6. Fungsi Objektif : Menuju Base musuh dan bersiap untuk menyerangnya saat ia pulang.

3.2. Eksplorasi Alternatif Solusi *Greedy*

Terdapat beberapa alternatif solusi *greedy* yang dilakukan oleh program bot ini. Tujuan dari permainan ini adalah memperoleh poin sebanyak banyaknya dengan mengumpulkan diamond semaksimal mungkin. Untuk mengumpulkan diamond semaksimal mungkin dapat dilakukan dengan meminimalisir jarak bot dengan diamond yang akan diambil sehingga bot lebih efisien dan efektif, dan juga dapat men-*tackle* bot lawan dan mengambil diamond yang dimilikinya, menggunakan *red button* untuk mengubah distribusi diamond pada board, dan menggunakan *teleport* untuk meminimalisir jarak ke posisi tujuan yang dilakukan oleh bot.

1. Collecting Diamonds yang terdekat dengan Bot

   Strategi ini adalah mengumpulkan diamond sebanyak banyaknya dengan mengambil diamond yang paling dekat dengan bot setiap langkah yang dilakukan oleh bot. Setelah diamond yang diperoleh oleh bot sudah 5, maka bot akan balik ke base. Namun, kadang strategi ini kurang efektif apabila setelah bot mendapatkan 5 diamonds, namun jarak untuk balik ke base sangat jauh sehingga membutuhkan waktu yang lama dan terkadang hingga waktu habis dan tidak sempat untuk ke base dan menambah point.

2. Collecting Diamonds yang terdekat dengan Base

   Strategi ini adalah mengumpulkan diamond sebanyak banyaknya namun hanya di sekitar base, kemudian setelah memperoleh 5 diamond, maka bot akan balik

ke base. Strategi ini digunakan agar waktu yang dibutuhkan setelah memperoleh 5 diamonds dan balik ke base lebih cepat sehingga point yang diperoleh bertambah.

3. Chasing Enemy

   Strategi ini adalah mengejar bot musuh yang terdekat dengan bot. Bot menyerang musuh yang terdekat dan memiliki jumlah diamonds jika dikali 5 lebih besar daripada jarak diamond terdekat dengan bot. Sehingga memungkinkan untuk men-*tackle* bot musuh tersebut dan memperoleh diamonds yang dimilikinya.

4. Attack Enemy

   Strategi ini adalah menyerang bot musuh dengan meninjau apakah ada bot musuh yang selisihnya satu satuan dengan bot kita, baik secara horizontal maupun vertikal. Apabila bot musuh memiliki posisi sumbu-x yang sama dengan bot kita atau posisi bot musuh dengan bot kita atas-bawah, maka kita menggeser bot ke kanan terlebih dahulu kemudian menggeser ke kiri, ke posisi semula agar dapat menghindar sekaligus men-*tackle* bot musuh tersebut apabila dia bergerak secara vertikal. Dengan cara yang sama juga apabila bot musuh memiliki sumbu-y yang sama dengan bot kita atau posisi bot musuh dengan bot kita bersebelahan, maka kita menggeser bot ke atas terlebih dahulu kemudian menggeser ke bawah lagi. Strategi ini sekaligus bisa menghindar dari lawan dan men-*tackle* lawan.

5. Defend

   Strategi bertahan yang dilakukan apabila terdapat bot musuh yang jaraknya satu satuan dengan bot kita baik horizontal maupun vertikal. Apabila ada bot musuh yang jaraknya satu satuan secara vertikal atau atas-bawah, maka dalam langkah itu, bot kita bergerak ke kanan dengan asumsi pergerakan dari bot musuh adalah vertikal, sehingga kita terhindar dari *tackle* bot musuh. Dengan cara yang sama juga, apabila ada bot musuh yang jaraknya satu satuan secara horizontal atau bersebelahan, maka dalam langkah itu, bot kita bergerak ke atas dengan asumsi pergerakan dari bot musuh adalah horizontal sehingga kita dapat menghindar juga dari *tackle* bot musuh.

6. Using Teleport untuk balik ke Base

Strategi ini digunakan untuk meminimalisir jarak bot untuk balik ke *base*. Saat jumlah diamonds yang diperoleh oleh bot sudah 5, maka bot akan balik ke *base*. Namun, terkadang jarak balik ke *base* sangat jauh. Tetapi, ada teleport yang bisa digunakan. Sehingga, saat diamonds yang dimiliki bot sudah 5 dan bot akan balik ke *base*, bot akan mengecek dahulu posisi teleport pada *board*, kemudian bot akan mengecek apakah jika menggunakan teleport akan lebih sedikit jarak yang akan ditempuh atau tidak, dengan mengecek jumlah jarak dari teleport terdekat dengan bot dan jarak dari pasangan teleport tersebut dengan base. Apabila jumlahnya lebih kecil dari jarak bot dengan *base*, maka bot dapat menggunakan teleport untuk balik ke *base*, sehingga waktu yang dibutuhkan akan lebih cepat.

7. Using Red Button

   Strategi ini digunakan ketika tidak ada diamond yang dekat dengan bot, ataupun base, kalau pun ada, bot harus melalui jarak yang cukup jauh untuk mendapatkan diamond. Kemudian mengecek posisi button pada *board*. Apabila jarak bot dengan button tersebut lebih kecil dibanding jarak bot dengan diamond atau dengan kata lain nilai minimum dari jarak diamond terdekat dengan base dengan jarak diamond terdekat dengan bot, dan nilai minimum ke diamond tersebut lebih dari 5, maka bot bergerak ke *red button* sebagai *goal position* agar distribusi diamonds berubah sehingga bot akan lebih efektif dan efisien dalam memperoleh diamond dalam waktu yang tersisa.

8. Collect & Chase

   Strategi ini merupakan kombinasi strategi collecting diamonds terdekat dari bot & base, chasing enemy, dan using teleport. Pada setiap langkah, bot akan memeriksa apakah diamond yang terdekat lebih dekat jaraknya dibandingkan menyerang musuh yang mungkin memiliki lebih dari 1 diamond. Teleport digunakan agar bot dapat kembali ke base secara lebih efektif.

9. Full Tackle

   Strategi ini merupakan kombinasi antara strategi Attack dan Defend. Bot akan terus mengejar siapapun musuh yang lokasinya terdekat. Apabila ia sudah dekat dengan musuh, yaitu selisih jaraknya sama dengan 1, ia akan melakukan aksi selanjutnya berdasarkan banyaknya diamond yang dimiliki musuh. Apabila diamond musuh lebih dari dirinya, maka ia akan melakukan Attack yaitu tackle dan mencuri diamond musuh. Sedangkan apabila dia memiliki diamond yang

kurang dari bot, ia akan melakukan defend yaitu menghindari tackle musuh dengan cara bergerak ke arah yang berbeda.

10. Begal

    Strategi ini merupakan strategi yang berbeda dengan lainya, karena ia selalu berada disekitar base lawan, lalu menyerangnya sebelum mereka menyimpan diamond di base. Bot akan mencari siapakah musuh yang memiliki banyak diamond terbanyak, lalu ia akan berjalan menuju base musuh tersebut. Bot kemudian berjalan mengelilingi bos lawannya sehingga ia siap untuk men-takcle lawan. Setelah berhasil takcle, ia akan pulang kembali ke basenya.

3.3. Analisis Efisiensi dan Efektivitas dari Kumpulan Alternatif Solusi *Greedy*

Setiap strategi pasti memiliki kelebihan dan kekurangan dalam segi efisiensi maupun efektivitasnya. Sehingga perlu dipertimbangkan lagi untuk memilih strategi mana yang akan digunakan dalam program bot. Untuk menganalisis efisiensi dan efektivitas strategi-strategi tersebut, akan ditinjau dari beberapa kriteria berikut:

- Jumlah diamond yang dikumpulkan dan dikembalikan oleh bot
- Waktu yang dibutuhkan oleh bot untuk mengumpulkan dan mengembalikan diamond ke base
- Risiko yang dihadapi oleh bot dari serangan musuh
- Keuntungan yang diperoleh oleh bot dari mengubah distribusi diamond atau menggunakan teleport

Berikut adalah analisis untuk setiap strategi:

1. Collecting Diamonds yang terdekat dengan Bot

   Strategi ini memiliki efisiensi yang tinggi, karena bot dapat menghemat waktu dan energi dengan mengambil diamond yang paling dekat dengan posisinya. Namun, strategi ini memiliki efektivitas yang rendah, karena bot tidak mempertimbangkan lokasi base atau musuh. Bot dapat kehilangan diamond yang sudah dikumpulkan jika diserang oleh musuh atau terjebak dalam jarak yang jauh dari base. Bot juga dapat melewatkan diamond yang lebih banyak atau lebih dekat dengan base jika hanya fokus pada diamond yang terdekat dengan dirinya. fungsi ini melakukan iterasi melalui semua intan di papan permainan, yang memiliki kompleksitas waktu O(n).

2. Collecting Diamonds yang terdekat dengan Base

Strategi ini memiliki efisiensi yang rendah, karena bot dapat menghabiskan waktu dan energi dengan mengabaikan diamond yang jauh dari base. Namun, strategi ini memiliki efektivitas yang tinggi, karena bot dapat meminimalkan risiko kehilangan diamond dan memaksimalkan poin yang diperoleh. Bot dapat mengembalikan diamond dengan cepat dan aman ke base, dan mengambil diamond yang lebih mudah dijangkau jika ada. fungsi ini melakukan iterasi melalui semua intan di papan permainan, yang memiliki kompleksitas waktu O(n).

3. Chasing Enemy

   Strategi ini memiliki efisiensi yang rendah, karena bot dapat mengorbankan waktu dan energi dengan mengejar musuh yang mungkin bergerak secara acak atau berlindung. Namun, strategi ini memiliki efektivitas yang tinggi, karena bot dapat meningkatkan poin dengan merebut diamond dari musuh dan mengurangi poin musuh dengan menghancurkan bot mereka. Bot dapat memanfaatkan keunggulan jumlah diamond atau jarak diamond untuk menyerang musuh yang lemah. Fungsi ini melakukan iterasi melalui semua bot musuh di papan permainan, yang memiliki kompleksitas waktu O(m), di mana m adalah jumlah bot musuh.

4. Attack Enemy

   Strategi ini memiliki efisiensi yang sedang, karena bot dapat menghemat waktu dan energi dengan menyerang musuh yang berdekatan dengan dirinya. Namun, strategi ini memiliki efektivitas yang sedang, karena bot tidak mempertimbangkan jumlah diamond atau jarak base dari musuh. Bot dapat menyerang musuh yang tidak memiliki diamond atau yang dekat dengan base mereka, sehingga tidak mendapatkan keuntungan yang besar. Bot juga dapat terjebak dalam pertempuran yang berkepanjangan dengan musuh yang memiliki diamond atau yang jauh dari base mereka, sehingga berisiko kehilangan diamond atau waktu. Fungsi ini melakukan iterasi melalui semua bot musuh di papan permainan, yang memiliki kompleksitas waktu O(m), di mana m adalah jumlah bot musuh.

5. Defend

   Strategi ini memiliki efisiensi yang tinggi, karena bot dapat menghindari serangan musuh dengan gerakan yang minimal. Namun, strategi ini memiliki efektivitas yang rendah, karena bot tidak berusaha mengumpulkan atau mengembalikan diamond. Bot hanya bertahan di posisinya, sehingga tidak mendapatkan poin atau mengurangi poin musuh. Bot juga dapat menjadi sasaran empuk bagi musuh yang lebih agresif atau cerdik. Fungsi ini melakukan iterasi melalui semua bot musuh di papan permainan, yang memiliki kompleksitas waktu O(m), di mana m adalah jumlah bot musuh.

6. Using Teleport untuk balik ke Base

Strategi ini memiliki efisiensi yang tinggi, karena bot dapat mempercepat proses pengembalian diamond dengan menggunakan teleport. Strategi ini memiliki efektivitas yang tinggi juga karena dapat menghemat waktu untuk ke base dan menambah poin dengan cepat. Kompleksitas waktu fungsi ini didominasi oleh iterasi dan pencarian objek teleport, yaitu O(k), di mana k adalah jumlah objek game.

7. Using Red Button

   Strategi ini memiliki efisiensi yang rendah, karena bot dapat menghabiskan waktu dan energi dengan bergerak ke red button yang mungkin jauh dari posisinya. Namun, strategi ini memiliki efektivitas yang tinggi, karena bot dapat mengubah distribusi diamond yang mungkin tidak menguntungkan bagi dirinya. Bot dapat mendapatkan diamond yang lebih banyak atau lebih dekat dengan base dengan menekan red button, dan mengurangi diamond yang dimiliki oleh musuh atau yang jauh dari base mereka. Fungsi ini melakukan iterasi melalui semua objek game di papan permainan untuk menemukan posisi *red button*, sehingga kompleksitas waktu pencarian ini sebagai O(k), di mana k adalah jumlah objek.

8. Collect & Chase

   Strategi ini memiliki efisiensi yang cukup rendah namun lebih baik dibandingkan chasing enemy. Karena bot dapat memilih antara mengambil diamond dibandingkan hanya mengejar musuh saja. Tetapi banyaknya perhitungan yang perlu dilakukan dapat membuat bot berjalan lebih lambat sehingga peluang melakukan tackle pun berkurang. Terdapat resiko bahwa bot justru di tackle musuhnya terlebih dahulu, padahal seharusnya ia yang men-tackle musuh. Sedangkan efektivitas bot ini cukup tinggi karena mentackle musuh dapat memberikan diamond jauh lebih banyak dibandingkan hanya mengejar diamond. Fungsi ini melakukan iterasi terhadap semua bot musuh juga terhadap diamond, kompleksitas waktunya adalah O(m+n), dengan m banyaknya bot di arena dan n adalah banyaknya diamond.

9. Full Tackle

   Strategi ini memiliki efisiensi yang cukup tinggi, karena bot menghemat waktu dan energi dengan menyerang musuh yang berdekatan dengan dirinya. Terlebih bot juga menghindari serangan musuh dengan gerakan yang minimal. Namun, efektivitas bot ini cukup rendah. Seharusnya bot bisa memperoleh banyak diamond karena ia selalu mencuri diamond lawan dan menghindari serangan musuh. Pada kenyataannya, bot ini terlalu banyak menghabiskan waktu untuk mengejar musuh, terlebih ia tidak mengumpulkan diamond dengan sendirinya. Sehingga pada akhirnya diamond yang diperoleh sangat rendah. Karena hanya dilakukan 1 iterasi, kompleksitas waktunya adalah O(m) dengan m banyaknya bot.

10. Begal

Strategi ini memiliki efisiensi yang sedang, karena ia tidak banyak memerlukan perhitungan tetapi perlu berjalan menuju base musuh yang mungkin cukup jauh. Namun, secara efektivitas bot ini sangat rendah. Strategi ini tidak hanya menghabiskan banyak waktu akibat berjalan menuju base musuh, tetapi ia juga harus menunggu musuh tersebut pulang terlebih dahulu. Realitanya, bot sering kali gagal men-tackle musuh walaupun sudah di depan base, karena dengan bot selalu bergerak mengelilingi base bisa saja ia melewati kesempatan menyerang musuh, apalagi jika ia justru yang terkena tackle.

3.4. Strategi Greedy yang Digunakan dalam Program Bot

Strategi yang digunakan oleh program bot ini yaitu *Collecting Diamonds* yang terdekat dengan bot*, Collecting Diamonds* yang terdekat dengan base, *Using Teleport,* dan *Using Red Button*.

Pertama-tama, strategi fokus pada tujuan utama, yaitu mengumpulkan 5 diamonds untuk memenangkan permainan. Jika bot telah mengumpulkan 5 diamonds, strategi *Using Teleport* mengevaluasi apakah bot dapat menggunakan teleport untuk ke *base* atau tidak. Jika ternyata jarak yang ditempuh untuk balik ke *base* lebih efektif dengan *teleport*, maka gunakan *teleport*. Jika ternyata lebih jauh jika menggunakan *teleport*, maka *goal position* bot langsung diarahkan ke *base*.

Di sisi lain, jika jumlah diamonds belum mencapai 5, strategi mengevaluasi waktu yang tersisa dalam permainan. Jika waktu tersisa kurang dari 7000 milliseconds dan terdapat diamond yang jaraknya sekitar kurang dari atau sama dengan 2 satuan dari *base*, maka bot dapat mengambil diamond itu terlebih dahulu kemudian balik ke base, namun jika tidak ada *goal position* diarahkan ke *base*, untuk kembali ke base secara cepat. Jika waktu tersisa kurang dari 15000 milliseconds dan lebih dari atau sama dengan 7000 milliseconds dan terdapat diamond yang jaraknya sekitar kurang dari atau sama dengan 5 satuan dari *base*, maka bot dapat mengambil diamond itu terlebih dahulu kemudian balik ke base, namun jika tidak ada *goal position* diarahkan ke *base*, untuk kembali ke base secara cepat.

Terakhir, jika waktu tersisa lebih dari 15000 milliseconds, strategi mengevaluasi apakah menggunakan *red button* atau tidak. Jika posisi diamonds yang sudah ditinjau dari jarak diamond terdekat dengan bot dengan jarak diamond terdekat dengan base kebanyakan jauh, maka menggunakan *red button* yang akan mengubah distribusi diamonds pada *board* sehingga memudahkan bot untuk mengambil diamond secara efektif karena bisa saja posisinya akan lebih dekat dan jumlah diamonds lebih banyak di sekitar bot, namun jika tidak perlu menggunakan *red button*, bot akan mengambil diamond yang terdekat dengan bot. Dengan demikian, strategi ini mencerminkan strategi yang dinamis dan adaptif sesuai dengan kondisi permainan yang berubah-ubah.

BAB 4
Implementasi dan Pengujian

4.1. Implementasi Algoritma *Greedy*

Berikut adalah implementasi algoritma Greedy yang dilakukan bot dalam *pseudocode*. Strategi diimplementasikan menggunakan bahasa Python dalam file myBot.py

```
function initialize(self: myBot) -> myBot
{Menginisialisasi myBot}
        self.directions <- [atas, kanan, kiri, bawah]
        self.goal_position <- None
        self.current_direction <- 0
        -> self
```

```
function next_move(self: myBot, board_bot: GameObject, board: Board) -> direction
{Menentukan langkah terbaik selanjutnya secara Greedy}
        props <- board_bot.properties
        current_position <- board_bot.position
        base <- board_bot.properties.base

        {Jika diamonds sudah 5, pulang ke base}
        if (diamond yang kita koleksi = 5) then
            target_teleport_position, is_using_teleport <- self.using_teleport(board_bot, board)
            {Apabila lebih baik menggunakan teleport}
            if (is_using_teleport) then
                self.goal_position = target_teleport_position
            else
                self.goal_position = base

        else
            {Jika tidak ada tujuan spesifik, pilih langkah terbaik (greedy)}
            goal_from_base, min_distance_base <- self.nearest_diamond_from_base(board_bot, board)
            goal_from_player, min_distance_player <- self.nearest_diamond_from_bot(board_bot, board)
            min_distance <- minimal(min_distance_base, min_distance_player)
            goal_button, is_using_button <- self.using_button(board_bot, board, min_distance)

            {Apabila tersisa 7 detik terakhir, Langsung pulang ke base}
            if (waktu pada countdown < 7) then
                if (min_distance_base <= 2) then
                    self.goal_position <- goal_from_base
                else
                    self.goal_position <- base
            {Apabila interval waktu 7-15 detik tersisa, cari diamond terdekat dari base}
            else if (7 <= waktu pada countdown < 15) then
                if (min_distance_base <= 5) then
                    self.goal_position <- goal_from_base
                else
                    self.goal_position <- base
            {Apabila waktu tersisa > 15 detik, cari diamond terdekat dari base atau tekan button}
            else
```

```
                if (is_using_button) then
                    self.goal_position <- goal_button
                else
                    self.goal_position <- goal_from_player

        if (self.goal_position): then
            {Arahkan ke posisi tujuan}
            direction <- get_direction(
                current_position.x,
                current_position.y,
                self.goal_position.x,
                self.goal_position.y,
            )

        -> direction
```

```
function nearest_diamond_from_bot(self, board_bot: GameObject, board: Board) -> diamond,
distance
    {Jarak diamond terdekat dengan bot}
        diamonds <- set of diamonds
        if (diamond yang dikoleksi = 4) then
            blue_diamonds <- [diamond for diamond in diamonds if color is blue]
            if (number of blue diamonds > 0) then
                current_position <- bot current position
                distances <- [distance to every diamonds for diamond in blue_diamonds]
                nearest_diamond_index <- minimum in distances
                diamond_from_bot <- blue_diamonds[nearest_diamond_index].position
                min_distance_diamond_bot <- distances[nearest_diamond_index]
            else
                diamond_from_bot <- board_bot.properties.base
            return diamond_from_bot, min_distance_diamond_bot
        else
            diamonds <- [diamond for diamond in diamonds]
            current_position <- bot current position
            distances <- [distance to every diamonds for diamond in diamonds]
            nearest_diamond_index <- minimum in distances
            diamond_from_bot <- diamonds[nearest_diamond_index].position
            min_distance_diamond_bot <- distances[nearest_diamond_index]
            -> diamond_from_bot, min_distance_diamond_bot
```

```
function nearest_diamond_from_base(self, board_bot :GameObject, board: Board) -> diamond,
distance
{Jarak diamond terdekat dengan base}
        {Check for diamonds}
        base <- board_bot.properties.base
        min_distance_diamond_base <- 100000
        if (diamond yang dikoleksi = 4) then
            for gameObjects in board.game_objects do
```

```
                if (gameObjects type = "Diamond" and Color = blue) then
                    diamond_distance <- distance from diamond to bot
                    if (diamond_distance < min_distance_diamond_base) then
                        min_distance_diamond_base <- diamond_distance
                        diamond_from_base <- diamond position
            -> diamond_from_base, min_distance_diamond_base
        else
            for (diamond in board.diamonds) do
                diamond_distance = distance from diamond to bot
                if (diamond_distance < min_distance_diamond_base) then
                    min_distance_diamond_base <- diamond_distance
                    diamond_from_base <- position of diamond
            -> diamond_from_base, min_distance_diamond_base
```

```
function using_teleport(self, board_bot: GameObject, board: Board) -> teleport, boolean
{gunakan teleport saat pulang ke base (diamond sudah berjumlah 5)}
        is_using_teleport <- False
        base <- base properties
        Current_position <- bot position
        teleport_objects <- [List of empty position]

        for (game_object in board.game_objects) do
            if (game_object type = "Teleport") then
                append game object position to teleport_objects

        distance_teleport_first_from_bot <- calculate distance from first teleport to bot
        distance_teleport_second_from_bot <- calculate distance from second teleport to bot
        distance_teleport_first_from_base <- calculate distance from first teleport to base
        Distance_teleport_second_from_base <- calculate distance from second teleport to base
        distance_bot_from_base <- calculate distance from base to bot

        if (distance_teleport_first_from_bot <= distance_teleport_second_from_bot) then
            if (distance_teleport_first_from_bot + distance_teleport_second_from_base <=
distance_bot_from_base) then
                is_using_teleport <- True
                -> teleport, is_using_teleport
            else:
                ->None, is_using_teleport

        else:
            if (distance_teleport_second_from_bot + distance_teleport_first_from_base <=
distance_bot_from_base) then
                is_using_teleport <- True
                -> teleport, is_using_teleport
            else:
                -> None, is_using_teleport
```

```
function using_button(self, board_bot: GameObject, board: Board, min_distance: int) ->
button, boolean
{Menggunakan button jika posisi diamonds jauh dari bot sehingga memungkinkan bot memperoleh
```

```
diamond lebih dekat}
        current_position <-  current bot position
        button_position: [List of empty positions]

        for (game_object in board.game_objects) do
            if (game_object type = "Button") then
                append button position to button_position
                break

        Distance_bot_button <- calculate distance from bot to button

        if (min_distance > 5 and distance_bot_button < min_distance) then
            is_using_button <- True
        else
            is_using_button <- False

       -> button, is_using_button
```

4.2. Struktur Data Program

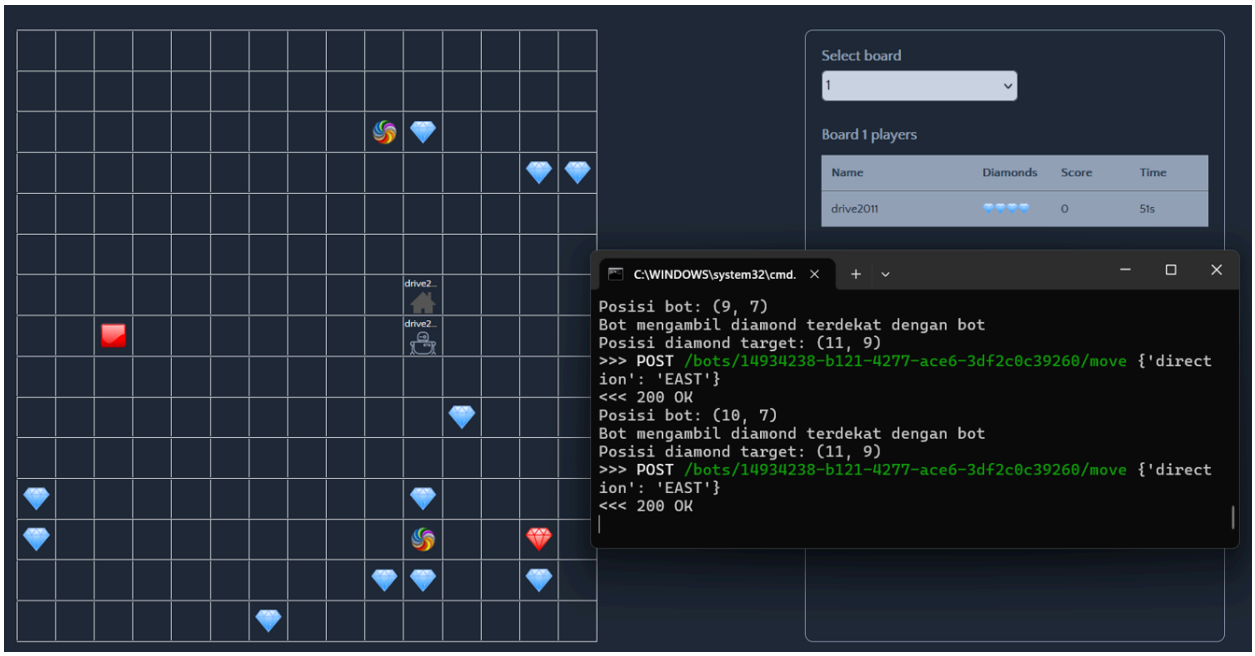
Struktur data program yang kami buat untuk bot permainan Diamonds kami adalah sebagai berikut.

```
src/
├── game/
│   ├── logic/
│   │   ├── unused/
│   │   │   ├── begal.py
│   │   │   ├── collectChase.py
│   │   │   └── fullTackle.py
│   │   ├── __init__.py
│   │   ├── base.py
│   │   ├── myBot.py
│   │   ├── random.py
│   │   └── testBot.py
│   ├── __init__.py
│   ├── api.py
│   ├── board_handler.py
│   ├── bot_handler.py
│   ├── models.py
│   └── util.py
├── decode.py
├── main.py
├── run-bots.bat
└── run_bot.sh
```

Seperti yang bisa dilihat pada struktur program diatas, terdapat 3 folder utama yang dijadikan modul dalam program, meliputi game, logic, dan unused. 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 myBot.py. Sedangkan folder unused berisi strategi yang telah kami implementasikan namun tidak kami gunakan sebagai bot utama.

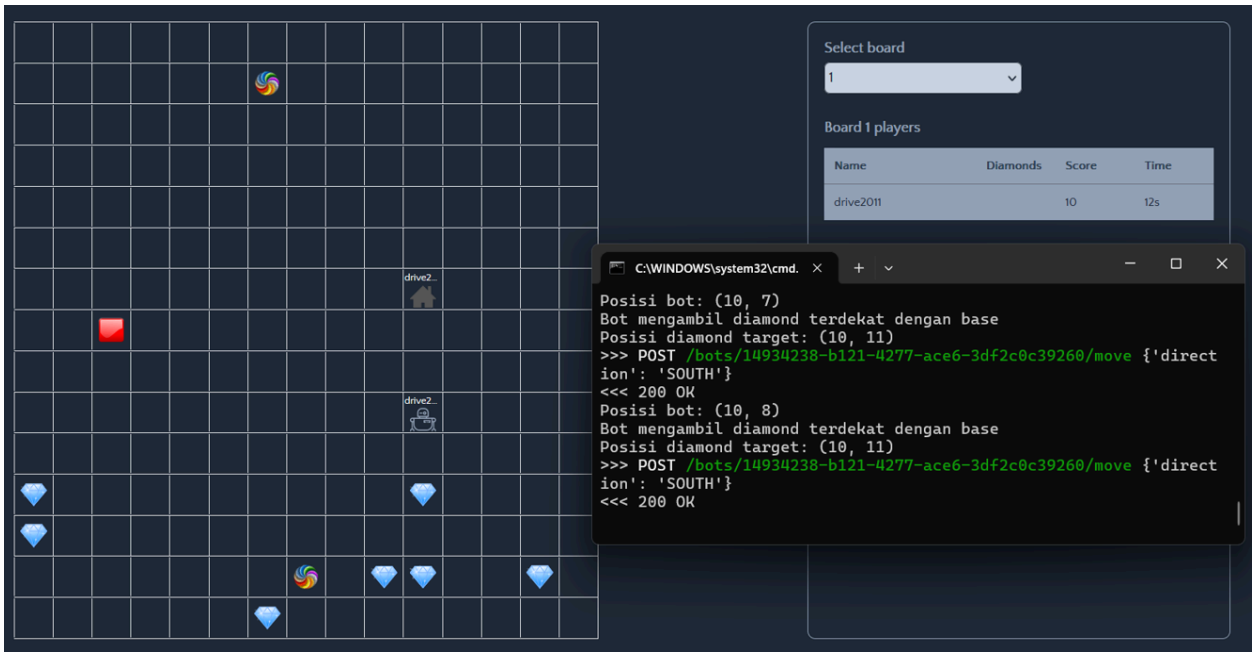
4.3. Analisis dan Pengujian

1. Collecting Diamonds yang terdekat dengan Bot



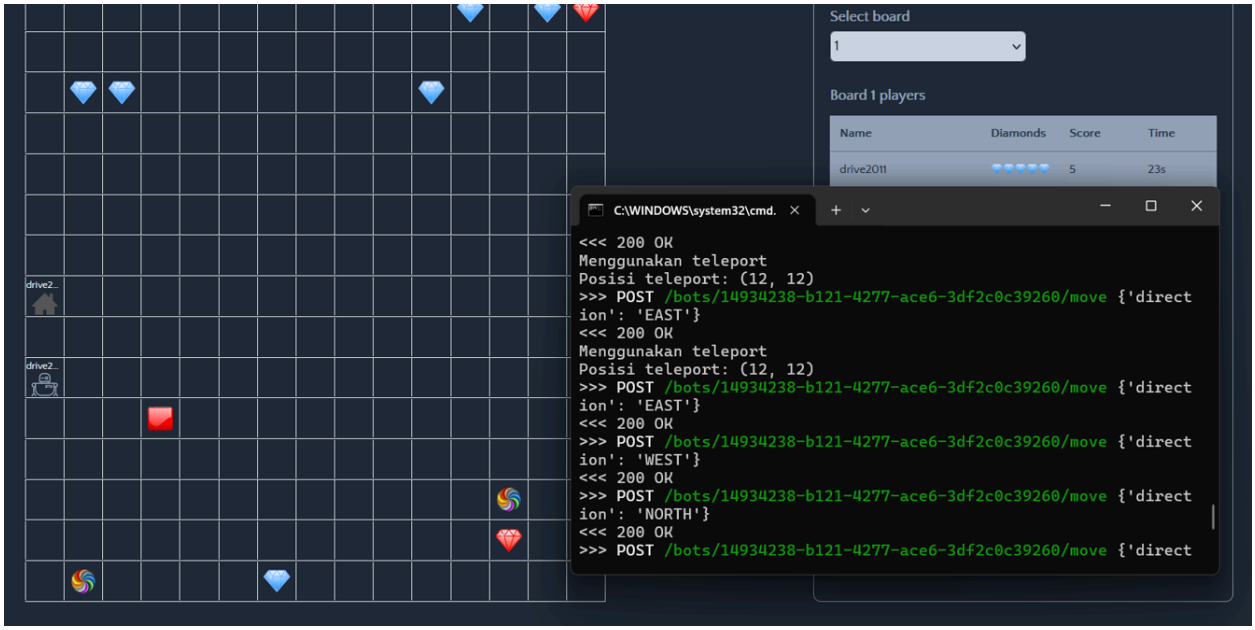
Dapat dilihat pada gambar di atas, posisi bot berada di titik (10,7) dan bot akan mengincar diamond. Karena waktu yang tersisa masih di atas 15 detik, maka bot masih cukup untuk mengambil diamond satu lagi dan diamond yang paling dekat dengan bot adalah diamond yang berada di titik (11,9). Hal ini sangat efektif untuk menambah jumlah diamond dalam waktu yang singkat.

2. Collecting Diamonds yang terdekat dengan Base



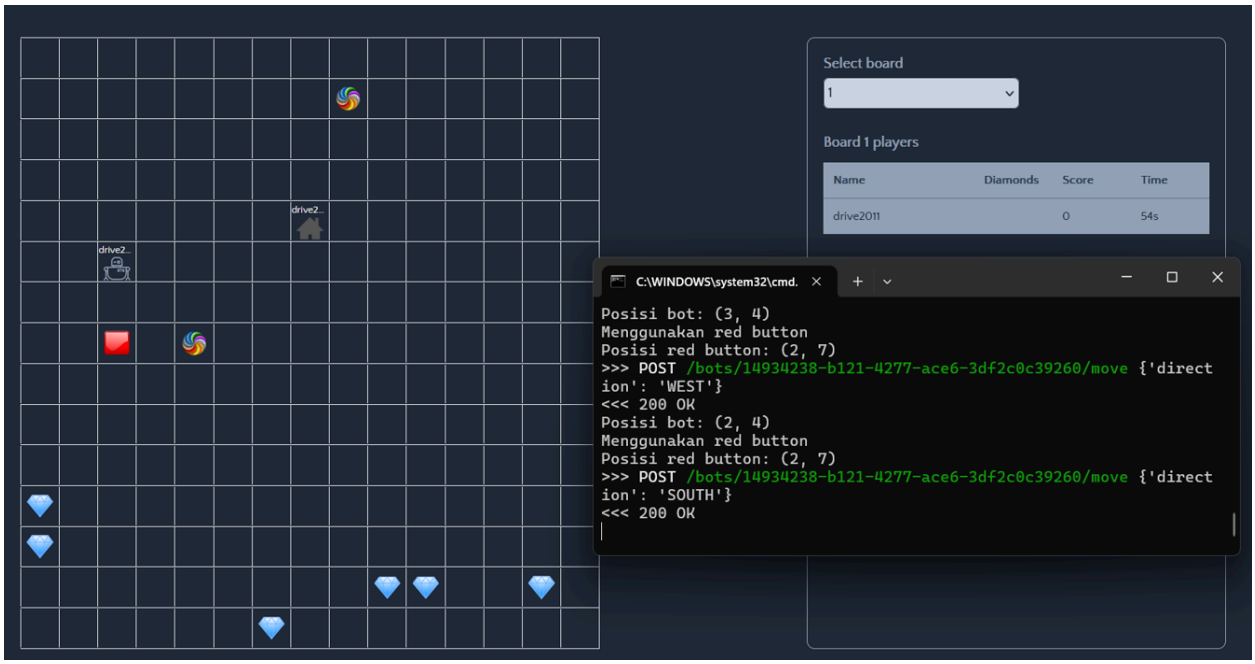
Pada gambar di atas, bot mengambil posisi diamond yang letaknya paling dekat dengan base, namun pada kondisi itu juga, diamond tersebut paling dekat dengan bot. Karena waktu yang tersisa hanya 12 detik, maka bot hanya dapat mengecek diamond yang letaknya paling dekat dengan base. Dan diamond yang letaknya paling dekat dengan base berada di posisi (10,11) saat bot berada di titik (10,8). Sehingga sangat dekat dan bot dapat mengambil diamond itu lalu balik ke base. Maka dalam hal ini, diamond dapat mengoptimalkan jumlah diamond yang mungkin bisa diambil walaupun waktu yang akan habis.

3. Using Teleport untuk balik ke Base



Pada gambar di atas, bot telah menggunakan teleport untuk balik ke base saat jumlah diamonds yang diperoleh oleh bot sudah 5, sehingga bot dapat balik ke base dengan cepat dan memperoleh poin dengan cepat.

4. Using Red Button



Dapat dilihat pada gambar di atas, bot berada di posisi (2,4). Namun tidak ada diamond yang dekat olehnya, maka bot dapat menggunakan red button yang posisinya di (2,7) yang dekat dengan bot. Sehingga, distribusi diamond akan berubah dan memungkinkan bot untuk mengumpulkan diamonds yang letaknya lebih dekat dengan bot dan dapat memaksimalkan jumlah diamonds yang dikumpulkan.

BAB 5
Kesimpulan dan Saran

5.1. Kesimpulan

Konsep algoritma greedy adalah algoritma yang berfokus pada pengambilan keputusan lokal yang optimal. Dengan mengkombinasikan dengan beberapa strategi tambahan lainnya, algoritma greedy mampu memberikan solusi yang optimal pada permainan Etimo Diamonds. Namun, solusi yang diberikan terkadang tidak optimum global mengingat ini memang adalah salah satu kelemahan algoritma greedy.

5.2. Saran

Beberapa saran yang kami dapatkan dari tugas besar kali ini adalah sebagai berikut:

1. Perlu dilakukan pemahaman mendalam terlebih dahulu terhadap kode permainan secara garis besar sehingga saat *debugging* jadi lebih gampang.
2. Perlunya penetapan terhadap strategi yang akan digunakan terlebih dahulu sehingga tidak membuang banyak waktu hanya untuk memikirkan strategi yang akan diimplementasikan.

Lampiran

Link *repository* github : https://github.com/randyver/Tubes1_Drive-2011
Link video youtube : https://youtu.be/H9cUgGn-zJ4?si=TDmEQHdzw-2u2x7e

Daftar Pustaka

Munir, Rinaldi. "Algoritma Greedy - Bagian 1." Institut Teknologi Bandung, 2020-2021. informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

Munir, Rinaldi. "Algoritma Greedy - Bagian 2." Institut Teknologi Bandung, 2020-2021. informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

Munir, Rinaldi. "Algoritma Greedy - Bagian 3." Institut Teknologi Bandung, 2021-2022. informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf

GeeksforGeeks. "Greedy Algorithms." Greedy Algorithms - GeeksforGeeks