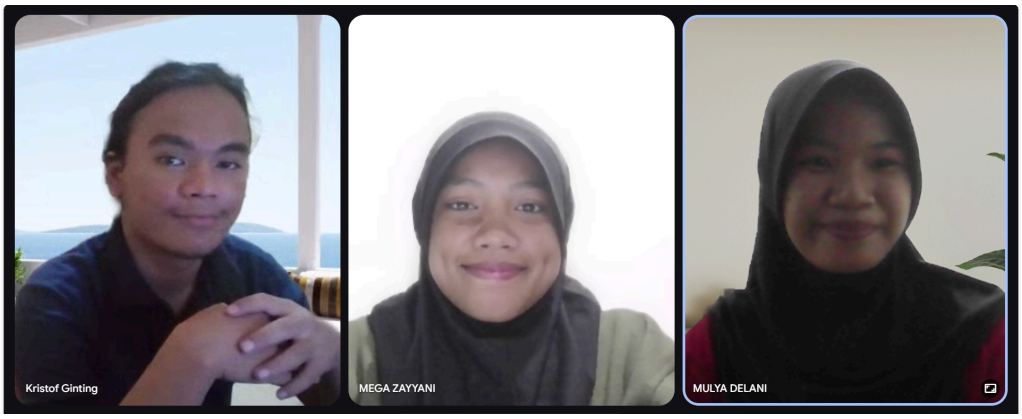
PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PERMAINAN DIAMONDS

Tugas Besar

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



Oleh: Kelompok 9 (STIMANANA)

Kristof Tsunami Ginting	123140117
Mulya Delani	123140019
Mega Zayyani	123140180

Dosen Pengampu: Imam Eko Wicaksono, S.Si., M.Si.

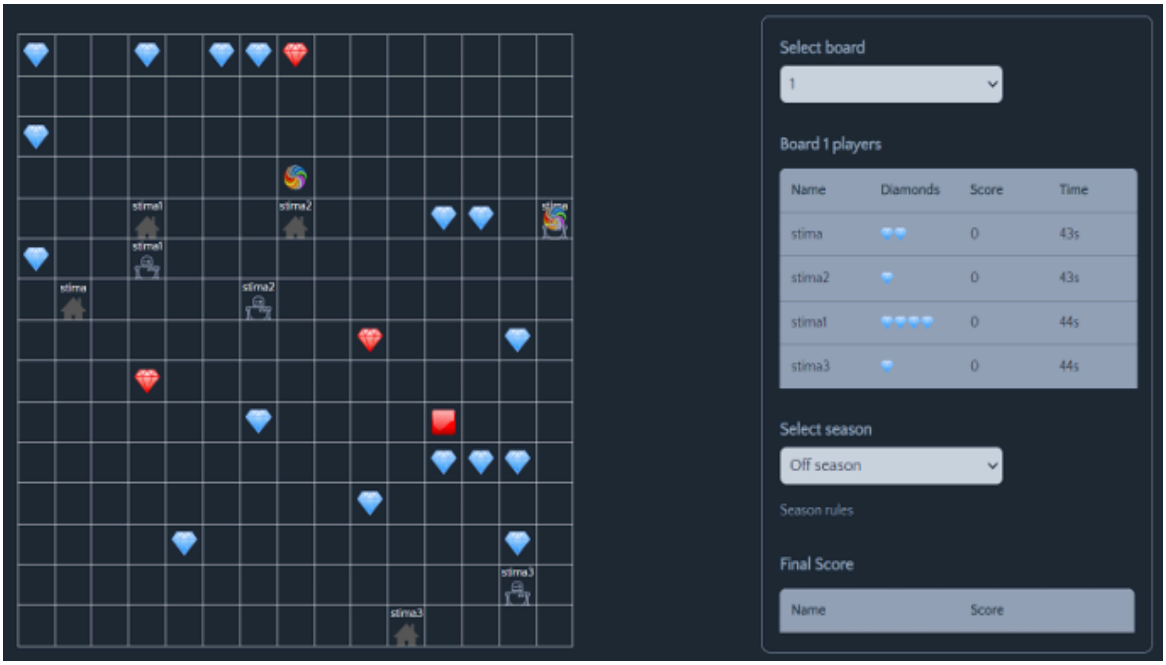
PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
2025

DAFTAR ISI

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.



Gambar 1. Ilustrasi Permainan Diamonds

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
    c. Program utama (main) dan utilitas lainnya.

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 dapat 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 II

LANDASAN TEORI

2.1 Dasar Teori

Algoritma *greedy* merupakan salah satu pendekatan penting dalam desain algoritma yang digunakan untuk menyelesaikan permasalahan optimasi. Pendekatan ini bekerja dengan membuat keputusan terbaik pada setiap langkah berdasarkan informasi yang tersedia saat itu, tanpa mempertimbangkan dampak keputusan tersebut di masa depan. Prinsip yang dipegang algoritma ini adalah *greedy choice property*, yaitu bahwa pilihan optimal lokal diharapkan mengarah pada solusi optimal global. Selain itu, masalah yang cocok diselesaikan dengan pendekatan *greedy* harus memiliki *optimal substructure*, yaitu solusi optimal dari permasalahan dapat dibentuk dari solusi optimal sub masalahnya. Keunggulan utama dari algoritma *greedy* adalah kesederhanaannya dalam implementasi serta efisiensinya dalam waktu komputasi. Namun, pendekatan ini tidak selalu menjamin solusi optimal, terutama pada masalah yang tidak memenuhi kedua sifat tersebut. Beberapa contoh penerapan algoritma *greedy* dalam bidang ilmu komputer antara lain adalah algoritma Kruskal dan Prim untuk minimum spanning tree, Huffman koding untuk kompresi data, serta *activity selection problem* dalam penjadwalan aktivitas [1],[2]

2.2 Cara Kerja Program

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

1. Program bot akan mengecek apakah bot sudah terdaftar atau belum, dengan mengirimkan POST request terhadap endpoint /api/bots/recover dengan body berisi email dan password bot. Jika bot sudah terdaftar, maka backend akan memberikan response code 200 dengan body berisi id dari bot tersebut. Jika tidak, backend akan memberikan response code 404.

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

2.3 Cara Implementasi Program

Cara kerja program ini adalah dengan diadakannya pengimplementasian terhadap *Artificial Intelligence* menggunakan pemanfaatan algoritma greedy untuk mengendalikan pergerakan pada bot dalam permainan ini. Pada logic ini bot akan mengevaluasi kondisi permainan setiap giliran dan akan langsung memilih aksi yang paling menguntungkan tanpa mempertimbangkan akibat dalam jangka panjang. Dalam logic permainan ini implementasi program dilakukan dalam kelas StimananaLogic yang merupakan turunan dari kelas BaseLogic. Fungsi utama pada program adalah next_move() yang akan dipanggil oleh engine permainan setiap giliran. Melalui pengurutan logika prioritas dalam fungsi utama, ketika bot tersebut membawa lebih dari atau sama dengan tiga diamond dan posisi musuh dekat maka bot program akan langsung kembali. Akan tetapi ketika

jumlah diamond yang digunakan kurang dari tiga, maka bot akan memilih untuk menekan tombol merah. Fungsi-fungsi pembantu yang digunakan dalam bot program ini ada get_closet_diamond, should_press_red_button, dan distance_with_teleporter. Semua ini disimpan dalam file pyhton logic bot bernama stimanana.py yang dijalankan dalam perintah pyhton main.py sesuai dengan framework yang kami gunakan.

2.4 Menjalankan Bot Program

Berikut adalah langkah-langkah dalam menjalankan bot program:

1. Menyiapkan File Program
   kami menyimpan file program logika bot dengan nama stimana.py ke dalam folder game/logic yang sudah disediakan di dalam struktur project. File ini berisi seluruh logika pergerakan bot, termasuk bagaimana cara bot memilih arah gerak, mengambil diamond, pulang ke base, dan sebagainya.
2. Memastikan semua import berjalan
   impor berbagai modul penting seperti GameObject, Board, dan Position, serta fungsi get direction yang digunakan untuk menghitung arah gerak. Ini perlu dilakukan agar program bot dapat berinteraksi dengan objek-objek yang ada di dalam permainan.
3. Mendaftarkan bot permainan
   pendaftaran bot ke dalam sistem permainan. Caranya dapat berbeda-beda tergantung dari framework game-nya, tapi biasanya dilakukan dengan menambahkan bot ini ke konfigurasi atau daftar bot yang akan dimainkan. Intinya, kami memastikan bot StimananaLogic dikenali oleh sistem saat game dijalankan.
4. Menjalankan bot permainan
   Dengan membuka terminal atau command prompt, lalu menjalankan perintah seperti python main.py atau perintah lain sesuai sistem yang digunakan. Setelah itu, permainan akan dimulai, dan sistem akan memanggil logika next_move() dari bot setiap giliran secara otomatis.
5. Analisa gerak bot
   Dapat melihat langsung bagaimana bot bergerak. Dimana bot akan mengambil diamond terdekat, menghindar jika ada musuh di sekitar, menggunakan teleporter jika lebih cepat, dan kembali ke base jika waktunya sudah mepet atau inventory sudah penuh.
6. Uji dan evaluasi
   Untuk memastikan logika bot dapat berjalan dengan sesuai, biasanya kami melakukan uji coba beberapa kali sehingga kami dapat melakukan evaluasi

apakah strategi greedy yang diterapkan sudah efektif atau masih adanya perbaikan yang harus dilakukan.

BAB III

APLIKASI STRATEGI GREEDY

3.1 Proses *Mapping*

Dalam permainan Diamonds, ada banyak objek penting seperti bot (pemain), base (markas), diamond (diamond), serta objek khusus seperti teleporter dan red button. Bot kami akan membaca kondisi papan permainan secara langsung setiap saat. Bot akan mencari diamond terdekat, melihat apakah masih dapat menyimpan diamond (inventory-nya belum penuh), dan juga memperhitungkan sisa waktu. Dari semua itu, bot akan memutuskan apakah harus terus mencari diamond atau kembali ke base. Bot juga dapat memilih rute tercepat dengan atau tanpa teleporter.

Berikut adalah elemen-elemen dari algoritma greedy yang digunakan dalam pembuatan bot permainan diamonds:

1. Himpunan Kandidat, semua langkah atau aksi yang mungkin dilakukan oleh bot pada setiap tiba giliran permainan.
2. Himpunan Solusi, subset dari himpunan kandidat yang memberikan hasil terbaik dalam pengumpulan diamonds.
3. Fungsi Solusi, digunakan untuk mengecek total diamond yang berada dalam inventory apakah sudah memenuhi kapasitas secara maksimal atau belum.
4. Fungsi Seleksi, digunakan untuk memilih diamond berdasarkan density nya atau diamond dengan jarak paling dekat dan poin paling tinggi.
5. Fungsi Kelayakan, untuk memeriksa apakah kandidat diamond sudah pasti dapat diambil oleh bot dengan dua pertimabangan dengan memastikan apakah langkah yang dipilih sudah memenuhi syarat. Pertimbangan pertama adalah nilai density dari bot utama terhadap target diamond harus lebih besar dari nilai density bot musuh. Pertimbangan kedua adalah waktu perjalanan mengambil diamond target dan kembali ke base apakah masih cukup atau tidak.
6. Fungsi Objektif, memaksimalkan jumlah poin yang didapat oleh bot selama permainan.

3.2 Eksplorasi Alternatif Solusi Greedy

3.2.1 Greedy by Escape

Ketika bot membawa minimal 3 poin dan mendeteksi musuh dalam jarak dekat (biasanya <= 2 blok), maka bot akan segera mundur dan kembali ke base. Strategi ini membantu menghindari kehilangan diamond yang sudah dikumpulkan, terutama jika musuh memiliki kemampuan untuk men-tackle. Strategi ini sangat berguna dalam permainan yang kompetitif, di mana banyak bot saling berebut dan menyerang satu sama lain.

3.2.2 Greedy by Return (Waktu)

Strategi ini akan aktif jika waktu yang tersisa dalam permainan sama dengan atau kurang dari jarak ke base. Bot akan langsung diarahkan untuk pulang ke base tanpa mencari diamond lagi. Hal ini bertujuan agar poin yang telah dikumpulkan tidak terbuang sia-sia karena waktu habis. Dengan strategi ini, bot memastikan dirinya tidak ketinggalan waktu, dan dapat mengamankan minimal sejumlah poin yang sudah ada di inventory.

3.2.3 Greedy by Return (Langkah)

Mirip dengan strategi sebelumnya, strategi ini memfokuskan pada jumlah langkah yang masih dapat diambil berdasarkan waktu tersisa. Jika sisa waktu hanya cukup untuk berjalan sampai base, maka bot akan langsung diarahkan untuk pulang. Strategi ini sangat penting dalam fase akhir permainan, karena mempertimbangkan jarak dan waktu secara bersamaan, dan menghindari risiko kehilangan semua diamond karena kehabisan waktu sebelum mencapai base.

3.2.4 Greedy by Tackle

Dalam kondisi tertentu, jika ada musuh dalam jarak 1 blok dan musuh tersebut membawa minimal 2 poin diamond, bot akan diarahkan untuk menyerang atau menabrak (tackle) musuh tersebut. Jika berhasil, semua diamond musuh akan berpindah ke inventory bot kita. Strategi ini sangat efektif untuk menaikkan skor dengan cepat, tetapi juga berisiko jika bot tidak memiliki jalan pulang yang cepat atau inventory yang sudah hampir penuh.

3.2.5 Greedy by Red Button

Digunakan ketika jumlah diamond di peta tinggal sedikit (kurang dari 10). Dalam kondisi ini, bot akan mengecek apakah jarak ke red button lebih dekat dibandingkan dengan diamond terdekat. Jika iya, maka bot akan menuju red button untuk mereset posisi dan jumlah diamond di peta. Strategi ini dapat mengubah arah permainan secara drastis, karena dapat menguntungkan bot yang sebelumnya kesulitan mencari target..

3.2.6 Greedy by Distance (Inventory Penuh)

Jika inventory sudah penuh (5 poin) atau waktu hampir habis, maka bot tidak akan lagi mencari diamond tambahan. Sebaliknya, bot akan langsung diarahkan untuk pulang ke base. Jika terdapat teleporter yang akan membuat perjalanan lebih cepat, maka bot akan menggunakan teleporter tersebut.

3.2.7 Greedy by Red Diamond

Karena diamond merah bernilai 2 poin (dibanding biru yang hanya 1 poin), strategi ini membuat bot lebih memilih mengejar diamond merah meskipun jaraknya sedikit lebih jauh. Strategi ini cocok digunakan saat inventory masih cukup dan diamond merah tidak terlalu sulit untuk dijangkau. Namun, karena diamond merah biasanya lebih langka dan

sering diperebutkan, bot rentan berada dalam posisi berisiko saat mencoba mengambilnya.

3.2.8 Greedy by Diamond

Jika inventory belum penuh dan tidak ada kondisi kritis seperti musuh dekat atau waktu hampir habis, maka bot akan mencari diamond terdekat, baik merah maupun biru. Strategi ini membantu menjaga alur permainan tetap aktif dan efisien. Diamond yang dipilih akan bergantung pada jarak dan nilai poin yang tersedia.

3.2.9 Greedy by Return (mampir ke base)

Saat bot sedang menuju lokasi lain (seperti diamond atau red button), tetapi jalurnya melewati base, maka bot akan mampir ke base terlebih dahulu jika sedang membawa diamond. Strategi ini sangat praktis untuk mengamankan poin tanpa perlu membuat rute pulang yang terpisah.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

Berikut adalah analisis dan efektivitas dari masing-masing strategi yang diterapkan:

- Greedy by Escape

  Strategi ini sangat efisien dalam menjaga diamond yang telah dikumpulkan. Dimana Ketika bot membawa lebih dari atau sama dengan tiga diamond dan musuh berada dalam jarak yang dekat, maka bot langsung kembali ke base untuk menghindari kehilangan diamond karena tackle. Ini mengurangi risiko kehilangan hasil pencarian dan mempercepat keputusan defensif secara otomatis.

- Greedy by Return (waktu)

  Strategi ini efektif dalam mencegah bot kehabisan waktu saat membawa diamond, sekaligus efisien karena memperhitungkan dua jalur secara langsung dan teleportasi untuk memilih yang tercepat dengan melakukan perbandingan terhadap waktu.

- Greedy by Return (Langkah)

  Kurang lebih sama dengan greedy by return dalam skala waktu, akan tetapi strategi ini berfokus pada jumlah langkah yang aktual daripada pada waktu yang berjalan.

- Greedy by Tackle

  Untuk meminimalkan risiko dan memberikan peluang untuk mendapatkan diamond tambahan dengan cepat. Namun, efektivitasnya tetap bergantung pada posisi relatif musuh.

- Greedy by Red Button

Untuk memanggil lebih banyak diamond ke arena permainan dengan menekan red button saat diamond berada di board pada posisi lebih dari sepuluh. Hal ini meningkatkan peluang bot untuk tetap produktif dalam mencari diamond saat supply rendah.

- Greedy by Distance

  Saat inventory penuh atau waktu hampir habis, maka bot langsung kembali. Sehingga efisiensinya tinggi karena tidak membuang waktu menjelajahi diamond yang sudah tidak dapat dikumpulkan lagi.
- Greedy by Red Diamond

  Jika bot sudah membawa lebih dari tiga diamond dan ada red diamond yang dekat, maka bot akan memilih mengambilnya terlebih dahulu sebelum kembali ke base untuk menyeimbangkan resiko.
- Greedy by Diamond

  Memprioritaskan diamond yang paling mudah dijangkau.
- Greedy by Return (Mampir ke base)

  Bot akan menyimpan diamond saat melewati base, meskipun sedang menuju tujuan lain untuk mengamankan progress.

Gabungan semua strategi ini membuat bot lebih fleksibel dan kuat menghadapi berbagai kondisi permainan juga efisien dalam pengambilan keputusan dan dalam memaksimalkan poin.

3.4 Strategi Greedy yang Dipilih

Dalam pengembangan bot permainan diamonds, strategi algoritma greedy dipilih karena kesederhanaan serta kemampuannya untuk memberikan solusi yang cepat dan cukup optimal dalam konteks permainan berbasis waktu nyata. Pada bot permainan kami yaitu StimananaLogic mengimplementasikan greedy by density sebagai strategi utama yang dipilih.Karena pada konsep ini bekerja dengan cara menghitung nilai *density* dari setiap diamond, yaitu rasio antara poin yang dapat didapatkan dari diamond dengan jarak bot menuju diamond tersebut. Saat semakin tinggi nilai density-nya, maka akan semakin tinggi prioritas diamond tersebut untuk diambil. Strategi ini memungkinkan bot untuk terus bergerak ke arah diamond yang paling menguntungkan dan efisien. Akan tetapi, untuk menghadapi berbagai kondisi dinamis dalam permainan, strategi utama ini didukung dengan sub strategi greedy adaptif yang mengatur respons bot terhadap situasi tertentu, antara lain:

a. Greedy by Escape, ketika membawa lebih dari tiga diamond dan ketika didekati musuh.
b. Greedy by Return, ketika waktu hampir habis;

c. Greedy by Tackle, untuk menyerang musuh yang membawa banyak diamond
d. Greedy by Red Button, ketika jumlah diamond di papan sedikit
e. Greedy by Red Diamond, untuk memaksimalkan poin
f. Greedy by Distance dan Greedy by Return (Mampir ke Base) untuk efisiensi langkah.

Sehingga ketika menggabungkan strategi utama dan sub strategi, maka bot akan dapat bergerak lebih fleksibel dengan tidak hanya fokus pada satu tujuan saja.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

4.1.1 Pseudocode

```
#Kristof Tsunami Ginting (123140117)
#Mulya Delani (123140019)
#Mega Zayyani (123140180)
from typing import Optional
import random
from game.logic.base import BaseLogic
from game.models import GameObject, Board, Position
from ..util import get_direction

class StimananaLogic(BaseLogic):
    def __init__(self):

        self.goal: Optional[Position] = None

    def distance(self, A: Position, B: Position):
        return abs(A.x - B.x) + abs(A.y - B.y)

    # Fungsi untuk mencari semua objek bertipe teleporter di board
       # Berguna jika nanti ingin menggunakan teleportasi untuk
efisiensi jarak
    def get_teleporters(self, board: Board):
           return [obj for obj in board.game_objects if obj.type ==
"TeleportGameObject"]



       # Fungsi untuk menghitung jarak antara dua titik dengan
mempertimbangkan teleport
        def distance_with_teleporter(self, start: Position, end:
Position, board: Board):
        teleporters = self.get_teleporters(board)
        if len(teleporters) < 2:
            return self.distance(start, end)
```

```
        entry, exit = teleporters
                via_tp  =  self.distance(start,  entry.position)  +
self.distance(exit.position, end)
        return min(self.distance(start, end), via_tp)

    def get_best_teleport_or_base(self, pos, base, board):
        teleporters = self.get_teleporters(board)
        if len(teleporters) >= 2:
            entry, exit = teleporters
        dist_normal = self.distance(pos, base)
                dist_tp  =  self.distance(pos,  entry.position)  +
self.distance(exit.position, base)
        if dist_tp < dist_normal:
            return entry.position
        return base

    # Fungsi untuk menghitung arah langkah menuju base bot
    def return_to_base(self, bot: GameObject):
        base = bot.properties.base
            dx = 1 if base.x > bot.position.x else -1 if base.x <
bot.position.x else 0
            dy = 1 if base.y > bot.position.y else -1 if base.y <
bot.position.y else 0
        return dx, dy

    # Fungsi untuk mencari diamond terdekat dari bot
      def get_closest_diamond(self, bot: GameObject, board: Board,
red_only=False):
        diamonds = []
        max_capacity = 5
        current = bot.properties.diamonds
        for d in board.diamonds:
            if red_only and d.properties.points == 2:
                if current + 2 <= max_capacity:
                    diamonds.append(d)
            elif not red_only and d.properties.points == 1:
                if current + 1 <= max_capacity:
                    diamonds.append(d)
        if not diamonds:
            return None
```

```
        closest = diamonds[0]
        min_dist = self.distance(bot.position, closest.position)
        for d in diamonds[1:]:
            dist = self.distance(bot.position, d.position)
            if dist < min_dist:
                closest = d
                min_dist = dist
        return closest

    # Fungsi untuk mencari musuh terdekat dalam radius tackle (jarak
2)
    # Digunakan dalam strategi Greedy by Tackle
    def find_enemy_to_tackle(self, bot: GameObject, board: Board):
        for enemy in board.bots:
              if enemy.id != bot.id and self.distance(bot.position,
enemy.position) == 2:
                return enemy.position
        return None

    # Fungsi untuk menemukan red button  di board
    # Digunakan untuk strategi Greedy by Red Button
    def get_red_button(self, board: Board):
        for obj in board.game_objects:
            if obj.type == "DiamondButtonGameObject":
                return obj
        return None

    # Fungsi untuk menentukan bot menekan red button saat diamonds
kurang dari 10
       def  should_press_red_button(self,  bot:  GameObject,  board:
Board):
        if len(board.diamonds) < 10:
            red_button = self.get_red_button(board)
              nearest_diamond = self.get_closest_diamond(bot, board,
red_only=True) or \
                                  self.get_closest_diamond(bot, board,
red_only=False)
            if red_button and nearest_diamond:
                          dist_button  =  self.distance(bot.position,
red_button.position)
```

```
                        dist_diamond = self.distance(bot.position,
nearest_diamond.position)
                if dist_button < dist_diamond:
                    return red_button.position
        return None



    # Fungsi utama bot untuk menentukan langkah selanjutnya
    def next_move(self, bot: GameObject, board: Board):
        props = bot.properties
        pos = bot.position
        base = props.base
        time_left = getattr(board, "time_left", 999)
        steps_to_base = self.distance(pos, base)

          # Greedy by Escape: Jika dekat bot lawan (jarak <=2) dan
bawa >=3 diamond, langsung lari ke base
        if props.diamonds >= 3:
            for enemy in board.bots:
                    if enemy.id != bot.id and self.distance(pos,
enemy.position) <= 2:
                    self.goal = base
                            direction = get_direction(pos.x, pos.y,
self.goal.x, self.goal.y)
                    return direction

          # Greedy by Return: Pulang jika waktu tersisa kurang dari
atau sama dengan jarak ke base (pakai teleporter jika lebih cepat)
        dist_normal = self.distance(pos, base)
        dist_tp = self.distance_with_teleporter(pos, base, board)
        min_dist = min(dist_normal, dist_tp)
        if time_left <= min_dist:
              self.goal = self.get_best_teleport_or_base(pos, base,
board)
               direction = get_direction(pos.x, pos.y, self.goal.x,
self.goal.y)
            return direction
```

```
        # Pulang jika waktu tersisa sama atau kurang dari langkah ke
base
        if time_left <= steps_to_base:
            self.goal = base
                direction = get_direction(pos.x, pos.y, self.goal.x,
self.goal.y)
            return direction

          # Greedy by Tackle: Serang musuh yang bawa ≥2 diamond dan
jarak 2
        target_enemy = None
        for enemy in board.bots:
                if enemy.id != bot.id and getattr(enemy.properties,
"diamonds", 0) >= 2:
                if self.distance(pos, enemy.position) == 1:
                    target_enemy = enemy.position
                    break
        if target_enemy:
            self.goal = target_enemy

        #Greedy by Red Button: Tekan red button
        red_button_pos = self.should_press_red_button(bot, board)
        if red_button_pos:
            self.goal = red_button_pos

           # Greedy by Distance: Jika inventory penuh (≥5 diamond),
langsung pulang
        elif props.diamonds >= 5 or time_left <= steps_to_base + 1:
            # langsung pulang (pakai teleporter jika efisien)
               self.goal = self.get_best_teleport_or_base(pos, base,
board)

         # Greedy by Red Diamond: Jika sudah bawa ≥3 diamond, pulang
atau ambil red diamond jika dekat
        elif props.diamonds >= 3:
                        red = self.get_closest_diamond(bot, board,
red_only=True)
            if red and self.distance(pos, red.position) <= 3:
                self.goal = red.position
            else:
```

```
                    # Greedy by Teleporter: Gunakan teleporter jika
efisien untuk pulang
                    self.goal = self.get_best_teleport_or_base(pos,
base, board)

        # Greedy by Diamond: Cari red diamond, jika tidak ada baru
cari blue diamond
        else:
                    red = self.get_closest_diamond(bot, board,
red_only=True)
                    blue = self.get_closest_diamond(bot, board,
red_only=False)
            # Jika ada red diamond dan blue diamond, cek jarak
            if red and blue:
                dist_red = self.distance(pos, red.position)
                dist_blue = self.distance(pos, blue.position)
                # Jika blue diamond lebih dekat, ambil blue diamond
dulu
                if dist_blue < dist_red:
                    self.goal = blue.position
                # Jika red diamond lebih dekat, cek apakah muat di
inventory
                elif props.diamonds + 2 <= 5:
                    self.goal = red.position
                else:
                    # Jika Tidak muat ambil red diamond, langsung ke
base
                    self.goal = base
            elif red:
                    # Hanya ada red diamond, cek apakah muat di
inventory
                if props.diamonds + 2 <= 5:
                    self.goal = red.position
                else:
                    self.goal = base
            elif blue:
                self.goal = blue.position
            else:
                self.goal = base
```

```
            # Greedy by Return: Mampir ke base saat dilewati dan bawa
diamond
            if self.goal != base and self.distance(pos, base) == 1 and
props.diamonds > 0:
                self.goal = base

            # Hitung arah gerakan menggunakan get_direction
                direction = get_direction(pos.x, pos.y, self.goal.x,
self.goal.y)

            return direction
```

4.2 Struktur Data yang Digunakan

Pada Program bot diamonds yang kami kembangkan, struktur data yang digunakan agar dapat merepresentasikan elemen penting dalam bot adalah sebagai berikut:

1. GameObject
   Kelas ini digunakan untuk merepresentasikan semua objek-objek yang ada dalam bot permainan. Termasuk diamond, red button, maupun teleportasi. Dalam setiap gameobject akan memiliki atribut penting seperti type, position, dan properties. Atribut tersebut digunakan untuk menyimpan informasi-informasi seperti id bot, jumlah diamond yang dibawa dan posisi bot saat bermain.

2. Position
   Salah satu struktur data yang fungsinya adalah untuk menyimpan koordinat posisi dalam bentuk x dan y. Dalam struktur ini hampir semua perhitungan dalam algoritma greedy menggunakan objek position.

3. Board
   Merupakan representasi dari keseluruhan peta permainan yang di dalamnya mencakup semua objek-objek pada permainan. Dalam hal ini mencakup bot musuh, diamond, red button dan teleportasi. Board memiliki atribut berupa board.diamond, board.bots, dan board.game_object yang dimana fungsinya adalah untuk bisa mendapatkan data dari

lingkungan permainan yang kemudian akan dilakukan proses lebih lanjut menggunakan konsep logika greedy.

4. Optional [Position]
   Ini berasal dari kamus yang ada dalam python yang digunakan untuk mendefinisikan bahwa suatu variabel mungkin saja tidak memiliki nilai sama sekali. Sebagai contoh dari kode program kami ada atribut bernama self.goal, dimana pada atribut tersebut menunjukan bahwa bot dapat mendapatkan sebuah posisi yang jelas.

5. List (Tipe Data)
   Merupakan suatu tipe data yang digunakan dalam berbagai fungsi. Misalnya pada program bot ini terdapat get_teleporters yang digunakan untuk mengembalikan daftar seluruh teleportasi dalam bentuk list dari gameobject.

6. Dictionary
   Dalam properties meskipun tidak terlihat dalam kode pada gameobject tetapi data ini sangat penting karena digunakan untuk mengevaluasi kondisi permainan yang berlangsung.

7. get_direction (Fungsi Utility)
   Bukan struktur data utama yang ada pada gameobject, akan tetapi fungsi ini dapat digunakan hampir setiap kali bot permainan perlu melakukan pergerakan untuk mencapai suatu titik yang dituju.

4.3 Pengujian Program

4.3.1 Skenario Pengujian

Untuk memastikan bahwa strategi greedy yang digunakan pada bot *Diamonds* bekerja dengan baik, maka dilakukan beberapa skenario pengujian. Skenario ini dibuat dengan berdasarkan berbagai situasi yang mungkin terjadi di dalam permainan, seperti saat waktu hampir habis, inventory bot sudah penuh, atau ketika bot sedang dikejar oleh musuh.
Berikut adalah beberapa skenario pengujian yang dilakukan:

No	Skenario	Strategi yang Diuji
1	Bot membawa 3 diamond dan musuh ada di sekitar	Greedy by Escape
2	Waktu tersisa sama dengan jarak ke base	Greedy by Return (waktu)
3	Inventory bot sudah penuh	Greedy by Distance

4	Musuh berjarak 1 langkah dan membawa 2 diamond	Greedy by Tackle
5	Red button lebih dekat daripada diamond, dan diamond di board tinggal sedikit	Greedy by Red Button
6	Red diamond dekat dan masih ada ruang di inventory	Greedy by Red Diamond
7	Red diamond jauh, blue diamond lebih dekat	Greedy by Diamond
8	Bot melewati base saat membawa diamond	Greedy by Return (mampir)

4.3.2 Hasil Pengujian dan Analisis

No	Skenario	Hasil	Analisis
1	Bot membawa 3 diamond, musuh dekat	Bot langsung pulang ke base	Strategi ini efektif mencegah bot kehilangan diamond jika diserang
2	Waktu tinggal sedikit, jarak ke base pas	Bot langsung kembali ke base	Bot dapat memperhitungkan waktu dengan baik agar tidak kehabisan waktu
3	Inventory penuh (5 diamond)	Bot pulang lewat teleporter	Bot memilih rute tercepat, menunjukkan efisiensi jalur
4	Musuh dekat dan bawa banyak diamond	Bot menyerang musuh	Strategi ofensif berjalan sesuai rencana, potensi menambah poin tinggi
5	Red button lebih dekat dan diamond tinggal sedikit	Bot menekan red button	Bot dapat memprioritaskan red button untuk menambah jumlah diamond di board
6	Red diamond dekat dan masih bisa dibawa	Bot mengambil red diamond dulu sebelum pulang	Strategi ini mengoptimalkan poin tanpa mengorbankan keamanan
7	Red diamond jauh, blue lebih dekat	Bot memilih blue diamond	Bot mampu memilih target yang lebih realistis dan aman
8	Bot lewat dekat base sambil bawa diamond	Bot mampir ke base	Strategi ini efisien, karena bot dapat sekalian mengamankan diamond yang dibawa

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada permainan diamonds ini, dapat disimpulkan bahwa pendekatan greedy terbukti cukup efektif dalam menangani permasalahan pengambilan keputusan secara cepat dalam permainan ini. Dimana bot mampu mengambil tindakan yang sesuai berdasarkan kondisi saat itu, seperti memilih untuk segera kembali ke base jika membawa banyak diamond, menyerang musuh jika menguntungkan, atau memanfaatkan teleportasi untuk efisiensi pergerakan. Strategi-strategi ini diterapkan dengan mempertimbangkan berbagai aspek, seperti jarak, jumlah diamond, sisa waktu, serta posisi musuh. Meskipun begitu, strategi greedy yang digunakan belum sepenuhnya optimal dalam semua kondisi. Akan tetapi, struktur data dan fungsi-fungsi yang digunakan dalam program ini telah mendukung proses pengambilan keputusan secara efisien, serta cukup mudah ketika akan dikembangkan lebih lanjut.

5.2 Saran

Untuk pengembangan lebih lanjut pada permainan diamonds ini, strategi greedy yang digunakan dapat diperluas dengan menambahkan elemen prediksi, agar bot tidak hanya mempertimbangkan situasi saat ini, tetapi juga dampak langkah ke depan. Interaksi dengan musuh juga dapat ditingkatkan, misalnya dengan memperkirakan arah gerakan musuh agar bot dapat menghindari atau mendekati dengan lebih strategis. Pengujian juga sebaiknya dilakukan pada lebih banyak variasi skenario permainan agar performa strategi dapat dinilai secara lebih menyeluruh.

LAMPIRAN

A. Repository Github

https://github.com/Christ204/TUBES-STIGMA_STIMANANANA.git

B. Video Penjelasan (link Youtube)

https://youtu.be/4BZNm_dPY80

C. Link Google Drive

https://drive.google.com/file/d/158ceeucb7iFAic1WqFEA9yJWZEGEfXKi/view?usp=sharing

DAFTAR PUSTAKA

[1] *GeeksforGeeks*, “Greedy Algorithms,”. Diakses Kamis, 29 Mei 2025. Tersedia pada: https://www.geeksforgeeks.org/greedy-algorithms/.

[2] *GeeksforGeeks*, “Greedy Algorithms General Structure and Application,”. Diakses Kamis, 29 Mei 2025. Tersedia pada: https://www.geeksforgeeks.org/greedy-algorithms-general-structure-and-applications/

[3] Munir, Rinaldi. “Algoritma Greedy (Bagian 1)”. Diakses Sabtu, 31 Mei 2025. Tersedia pada: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

[4] Algoritma GMunir, Rinaldi. “reedy (Bagian 2)”. Diakses Sabtu, 31 Mei 2025. Tersedia pada: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

[5] Algoritma GMunir, Rinaldi. “reedy (Bagian 3)”. Diakses Sabtu, 31 Mei 2025. Tersedia pada: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf