Analisis *Procedural Generation* untuk Aplikasi Graf dan Kombinatorika dalam Pembentukan Peta Dinamis dalam Permainan *Hades*

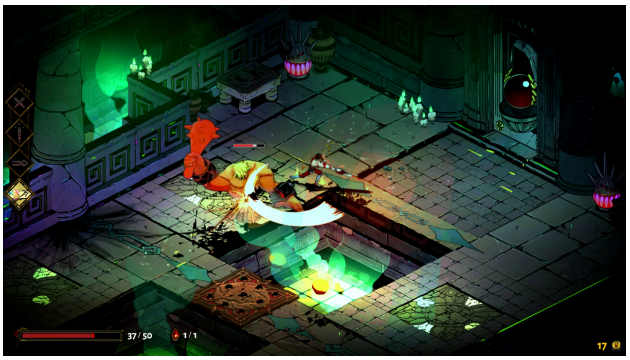
Ivant Samuel Silaban - 13523129[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523129@std.stei.itb.ac.id

Abstract*—Procedural Generation adalah teknik yang memungkinkan pembentukan elemen dalam permainan secara dinamis, menciptakan variasi dan pengalaman unik bagi pemain. Teknik ini sering digunakan pada permainan modern untuk meningkatkan replayability dan menantang pemain secara strategis. Dalam game Hades, teknik ini diterapkan untuk membangun peta dinamis di berbagai region seperti Tartarus, Asphodel, Elysium, hingga Temple of Styx. Makalah ini akan membahas bagaimana algoritma procedural generation berbasis graf dan teori kombinatorika digunakan untuk menciptakan peta dinamis dengan keragaman jalur, musuh, dan hadiah yang berbeda tergantung jalur yang dipilih. Dengan memanfaatkan struktur graf, setiap ruangan permainan dihubungkan secara fleksibel, sementara kombinatorika digunakan untuk menghitung kemungkinan jalur dan variasi musuh di setiap stage. Hasil implementasi menunjukkan bahwa loncatan ke stage lebih jauh membuat musuh lebih sulit, namun memberikan keuntungan strategis dengan mempercepat perjalanan menuju Final Boss. Integrasi sistem peluang pada Temple of Styx menambahkan tantangan unik bagi pemain dalam mencari kunci menuju pertarungan akhir. Penelitian ini menunjukkan bahwa procedural generation yang dirancang dengan baik dapat meningkatkan replayability dan memberikan pengalaman gameplay yang menarik melalui keragaman elemen.

***Keywords*—Graf, Hades, Kombinatorika, Peta Dinamis**

I. Introduction

Hades adalah game roguelike action yang dikembangkan oleh Supergiant Games. Pemain berperan sebagai Zagreus, putra Hades, yang berusaha melarikan diri dari dunia bawah menuju Gunung Olympus. Dalam perjalanan ini, pemain harus melewati empat region utama, yaitu Tartarus, Asphodel, Elysium, dan Temple of Styx, masing-masing diisi dengan tantangan berupa musuh dan bos di setiap akhir region. Pemain dapat memilih jalur yang berbeda di setiap permainan, mendapatkan bantuan dari para dewa Olympus, dan mengembangkan kemampuan untuk melanjutkan perjalanan. Salah satu daya tarik utama Hades adalah tingkat replayability yang tinggi berkat perubahan peta yang terjadi di setiap sesi permainan.



Gambar 1. Tampilan Permainan Hades

Peta dalam Hades dibangun menggunakan procedural generation, menciptakan pengalaman dinamis di mana susunan ruangan dan jalur antar ruangan berubah-ubah. Pendekatan ini memanfaatkan teori graf untuk merepresentasikan hubungan antar ruangan, sementara kombinatorika digunakan untuk menghitung variasi musuh yang mungkin muncul berdasarkan bobot total di setiap stage. Di Temple of Styx, probabilitas digunakan untuk menentukan lokasi kunci menuju Final Boss, menambahkan elemen ketidakpastian yang menarik.

Salah satu fitur gameplay utama adalah beragamnya hadiah yang diberikan tergantung jalur yang diambil oleh pemain. Pemain dapat memilih jalur berdasarkan reward tertentu, seperti tambahan nyawa atau kekuatan serangan, yang mempengaruhi strategi mereka. Selain itu, loncatan antar stage menawarkan jalur alternatif yang mempercepat perjalanan menuju Final Boss, namun meningkatkan tingkat kesulitan musuh sebagai konsekuensinya. Analisis ini bertujuan untuk menunjukkan bahwa prosedur ini tidak hanya menciptakan peta yang dinamis, tetapi juga memberikan pengalaman gameplay yang lebih menarik dan menantang melalui variasi jalur, musuh, dan hadiah.

II. Theoretical Basis

A. Graf

Graf adalah struktur matematika yang terdiri dari simpul (node) dan sisi (edge). Dalam procedural generation peta permainan Hades, graf digunakan

untuk merepresentasikan hubungan antar ruangan (simpul) dan jalur antar ruangan (sisi). Pada Hades, Graf yang digunakan adalah graf berarah (directed graph)

1. Graf Berarah

$G = (V, E)$

dimana:

- V adalah himpunan simpul
- $E \subseteq V \times V$ adalah himpunan sisi (edge) berarah

Misalnya, graf untuk chamber Tartarus memiliki struktur sederhana yaitu ruangan awal (simpul awal) memiliki sisi yang mengarah ke beberapa ruangan berikutnya.

Graf juga digunakan untuk merepresentasikan struktur Temple of Styx, di mana setiap ruangan adalah simpul, dan ruangan dengan kunci memiliki jalur langsung ke Final Boss.

B. Kombinatorika

Kombinatorika adalah cabang matematika yang mempelajari penghitungan pengaturan objek dalam suatu himpunan. Dalam konteks permainan Hades, kombinatorika digunakan untuk menentukan variasi musuh di setiap stage berdasarkan bobot total.

Formula kombinasi dengan penggantian (combinations with replacement):

$$C(n + r - 1, r) = \frac{(n+r-1)!}{r! \cdot (n-1)!}$$

Di mana:

n = jumlah jenis musuh (ringan, sedang, berat)

r = jumlah musuh dalam satu kombinasi, dihitung berdasarkan bobot total W.

C = Jumlah kemungkinan kombinasi

Setiap kombinasi memenuhi syarat:

$$\sum_{i=1}^{n} w_i \cdot x_i = W$$

dimana:

w_i = bobot jenis musuh ke-i

x_i = jumlah musuh jenis ke-i

W = total bobot musuh

Sebagai contoh, pada stage W = 15:

- Musuh ringan (w_i = 5), sedang (w_i = 10), dan berat (w_i = 15)
- Kemungkinan kombinasi: 3 x ringan, 1 x ringan + 1 x sedang, atau 1 x berat.

C. Peluang

Peluang adalah perluasan dari materi kombinatorika, namun untuk makalah ini hanya akan menggunakan satu rumus saja, yaitu:

$$P(A) = \frac{n(A)}{n(S)}$$

dimana,

$P(A)$ = Peluang

$n(A)$ = jumlah ruang sampel

$n(S)$ = jumlah ruang semesta kejadian

Teori peluang digunakan untuk menghitung probabilitas menemukan kunci di Temple of Styx. Setiap ruangan memiliki peluang yang sama untuk berisi kunci. Teori ini juga akan digunakan untuk menentukan berapa jumlah stage selanjutnya dan kemungkinan akan mendapat stage loncatan.

Untuk Temple of Styx, peluang awal untuk setiap ruangan adalah:

$$P(kunci\ ke\ ruangan\ 1) = \frac{1}{5}$$

Jika ruangan pertama tidak berisi kunci, peluang untuk ruangan kedua adalah:

$$P(kunci\ ke\ ruangan\ 2 \mid ruangan\ 1\ tidak\ ada) = \frac{1}{4}$$

Setiap kali ruangan diperiksa dan tidak ada kunci, peluang distribusi berubah menjadi:

$$P(kunci\ ruangan\ ke\ k \mid ke\ (k-1)\ tidak\ ada) = \frac{1}{5-(k-1)}$$

D. Peta Hades

1. Graf: Membentuk jalur antar ruangan dengan struktur logis dan bercabang.
2. Kombinatorika: Menghasilkan variasi musuh berdasarkan bobot total di setiap stage.
3. Peluang: Memberikan elemen acak dalam pencarian kunci di Temple of Styx dan elemen acak pada jalur stage pilihan.

III. Experiment

A. Simulasi Dalam Bahasa Python

Berikut adalah kode python dimana terdiri dari lima buah chamber, dimana chamber-chamber tersebut akan berjalan seperti berikut:

1. Chamber 1 (Tartarus)

Player akan masuk ke stage pertama, dan menghadapi beberapa musuh. Terdapat tiga musuh sebagai berikut:

Ringan = 5

Sedang = 10

Berat = 15

Pada stage pertama, total bobot musuh adalah 15, dan secara kombinatorik akan menghitung jumlah kemungkinan susunan

musuh yang ada pada stage tersebut dan setiap naik stage, bobot musuh akan bertambah juga. Pemain juga akan diminta inputan ingin lanjut ke stage selanjutnya atau tidak. Jika tidak, maka program berhenti, dan jika ya akan menampilkan daftar stage berikutnya. Dan dengan teori peluang, akan ada kesempatan dimana pemain bisa loncat ke dua stage selanjutnya. Contoh dari stage dua ke empat. Dan total stage dibatasi menjadi tujuh stages saja. Dan setiap lanjut stage, akan ada 2-3 pilihan stage selanjutnya + 0 - 2 pilihan stage lompatan. Dan jalur-jalur tersebut akan dibedakan dengan penambahan huruf a,b,c (contoh 2a, 2b,2c) untuk membedakan reward yang akan diterima, misalnya 2a akan mendapatkan reward a, 2b mendapatkan reward b, 2c akan mendapatkan reward c (pada permainan Hades sendiri memiliki variasi hadiah yang sangat banyak, tetapi simulasi ini akan menggunakan 3 variasi hadiah saja).
2. Chamber 2 (Asphodel)
Sama dengan Chamber 1 (Tartarus).
3. Chamber 3 (Elysium).
Sama dengan Chamber 1 (Tartarus).
4. Chamber 4 (Temple of Styx)
Pada game Hades, untuk ke chamber selanjutnya yaitu final boss, pemain harus menyingkirkan penghalang, yaitu Cerberus. Untuk menyingkirkannya, pemain harus mencari makanan Cerberus dari lima ruangan. Tetapi pada kode saya, saya akan menganggapnya dengan kunci. Pada chamber ini akan menggunakan teori peluang yang merupakan perluasan dari materi kombinatorika.
5. Chamber 5 (Final Boss - Hades).
Final Boss.
Berikut adalah kode tersebut:

```
import random

class HadesGame:
    def __init__(self):
        self.current_stage = 1
        self.current_chamber = 1
        self.stages_in_chamber = 0
        self.chamber_names = {
            1: "Tartarus",
            2: "Asphodel",
            3: "Elysium",
            4: "Temple of Styx",
            5: "Final Boss - Hades"
        }
        self.enemy_weights = {
            "ringan": 5,
            "sedang": 10,
            "berat": 15
        }
        self.stage_weight = 15
          self.MAX_STAGES_PER_CHAMBER = 7
          self.last_displayed_chamber = None

    def print_header(self):
        print("\n" + "="*50)
            print("SIMULASI PERMAINAN HADES".center(50))
        print("="*50)
        print("\nBobot Musuh:")
                for enemy, weight in self.enemy_weights.items():
                                print(f"-{enemy.capitalize()}: {weight}")
                print("\nSetiap chamber terdiri dari 7 stage")
                print("\nSetiap stage terdiri dari 2-3 pintu pilihan \nke stage berikutnya + 0 - 2 untuk meloncat \nke stage setelahnya")

        print("="*50 + "\n")

    def print_chamber_header(self):
         if self.last_displayed_chamber != self.current_chamber:
            print("\n" + "="*50)
                        print(f"Chamber {self.current_chamber} - {self.chamber_names[self.current_chamber]}".center(50))
            print("="*50)

self.last_displayed_chamber = self.current_chamber

                d                def generate_possible_enemy_combinations(self):
         enemies = ["ringan", "sedang", "berat"]
        possible_combinations = []
                    remaining_weight = self.stage_weight
```

```
                                    def
find_combinations(current_combo,
remaining_weight, start_idx):
            if remaining_weight == 0:

possible_combinations.append(tuple(sorte
d(current_combo)))
                return
            if remaining_weight < 0:
                return

             for i in range(start_idx,
len(enemies)):
                enemy = enemies[i]
                                    if
self.enemy_weights[enemy]           <=
remaining_weight:
                    find_combinations(
                        current_combo
+ [enemy],

remaining_weight                    -
self.enemy_weights[enemy],
                        i
                    )

                find_combinations([],
remaining_weight, 0)
                                  return
list(set(possible_combinations))


    def display_enemy_info(self):
        self.print_chamber_header()
                        combinations   =
self.generate_possible_enemy_combination
s()
                        print(f"\nStage
{self.stages_in_chamber  +  1}/7  (Total
Weight: {self.stage_weight})")
            print(f"Total possible enemy
combinations: {len(combinations)}")
```

```
                        selected_combo   =
random.choice(combinations)
        enemy_counts = {}
        for enemy in selected_combo:
                    enemy_counts[enemy]  =
enemy_counts.get(enemy, 0) + 1
            print("\nEncountered enemies
in this stage:")
                for  enemy,  count  in
enemy_counts.items():
                print(f"{count}x {enemy}
(Weight: {self.enemy_weights[enemy]})")

    def check_special_chamber(self):
        rooms = list(range(1, 6))
                            key_room    =
random.choice(rooms)
                    print(f"\nEntering
{self.chamber_names[4]}")
          print("Checking rooms for the
key...")

        for room in rooms:
                print(f"Checking room
{room}...")
            if room == key_room:
                print(f"Key found in
room {room}!")
                break

    def generate_next_stages(self):
                            can_jump    =
self.stages_in_chamber                  <
self.MAX_STAGES_PER_CHAMBER - 2
            offer_jump  =  can_jump  and
random.random() < 0.2

                    next_stage_num   =
self.stages_in_chamber + 2

                        next_stages   =
[f"{next_stage_num}{letter}"  for  letter
in  ['a',  'b',  'c'][:random.randint(2,
3)]]
```

```
            if offer_jump:
                jump_stage_num = min(next_stage_num + 1, self.MAX_STAGES_PER_CHAMBER)
                jump_stages = [f"{jump_stage_num}{letter}" for letter in ['a', 'b'][:random.randint(1, 2)]]
                next_stages.extend(jump_stages)

            return next_stages

        def play(self):
            self.print_header()
            while True:
                self.display_enemy_info()

                if self.stages_in_chamber == self.MAX_STAGES_PER_CHAMBER - 1:
                    print(f"\nBOSS FIGHT - {self.chamber_names[self.current_chamber]} Boss")
                    self.current_chamber += 1
                    self.stages_in_chamber = 0
                    self.stage_weight = 15

                    if self.current_chamber == 4:
                        self.check_special_chamber()

                    print(f"\nProceeding to {self.chamber_names[5]}...")
                    print("\nFinal Boss Fight with Hades!")
                    break
                continue

                continue_game = input("\nContinue to next stage? (y/n): ").lower()
                if continue_game != 'y':
                    break

                possible_stages = self.generate_next_stages()
                print("\nAvailable stages:", ", ".join(possible_stages))
                next_stage = input("Choose next stage: ")

                if next_stage in possible_stages:
                    stage_num = int(''.join(filter(str.isdigit, next_stage)))
                    stages_advance = stage_num - (self.stages_in_chamber + 1)
                    self.stages_in_chamber += stages_advance
                    self.stage_weight = 15 + (5 * self.stages_in_chamber)

    # Run the game
    game = HadesGame()
    game.play()
```

IV. Result

A. Header

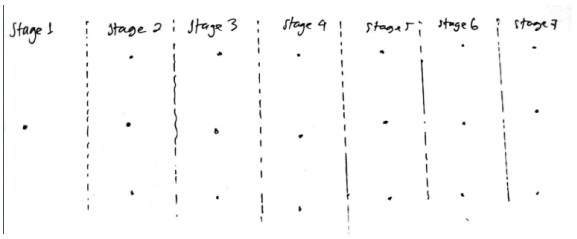
Berikut adalah output di awal permainan:

```
================================================
            SIMULASI PERMAINAN HADES
================================================

Bobot Musuh:
- Ringan: 5
- Sedang: 10
- Berat: 15

Setiap chamber terdiri dari 7 stage

Setiap stage terdiri dari 2-3 pintu pilihan
ke stage berikutnya + 0 - 2 untuk meloncat
ke stage setelahnya
================================================
```

Gambar 2. Output Header

B. Chamber

Graf Awal:



Gambar 2. Graf Awal Permainan untuk setiap Chamber

Notes:
Panah biasa: Jalur yang dapat dipilh (yang terlihat di program)
Panah tebal : Jalur yang dipilih
Panah putus-putus : Asumsi untuk jalur yang dapat dipilih namun tidak dipilih dapat memiliki jalur ke semua jalur di stage setelahnya
Titik di akhir: Boss

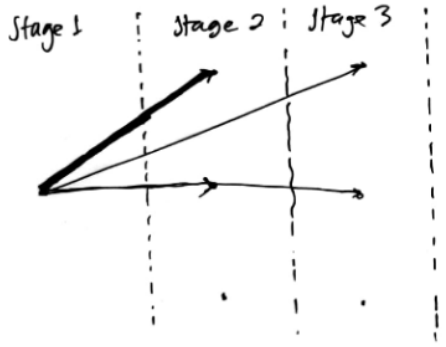
CHAMBER 1:

1. Stage 1

```
Stage 1/7 (Total Weight: 15)
Total possible enemy combinations: 3

Encountered enemies in this stage:
1x ringan (Weight: 5)
1x sedang (Weight: 10)

Continue to next stage? (y/n): y

Available stages: 2a, 2b, 3a, 3b
Choose next stage: 2a
```

Gambar 4. Output Stage 1

Graf Sementara:



Gambar 5. Graf Stage 1

Kombinatorika:

Output menunjukkan bahwa possible enemy combination adalah 3, dimana:

Weight = 15

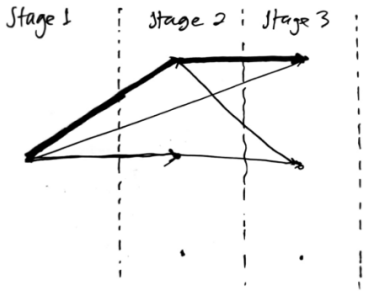
- ringan(5) → 10
  - ringan(5) → 5
    - ringan(5) → 0 ✓ [R,R,R]
  - sedang(10) → 0 ✓ [R,S]
- sedang(10) → 5
  - (skip ringan karena akan duplikat dengan [R,S])
- berat(15) → 0 ✓ [B]

2. Stage 2

```
Stage 2/7 (Total Weight: 20)
Total possible enemy combinations: 4

Encountered enemies in this stage:
1x berat (Weight: 15)
1x ringan (Weight: 5)

Continue to next stage? (y/n): y

Available stages: 3a, 3b
Choose next stage: 3a
```

Gambar 6. Output Stage 2

Graf Sementara:



Gambar 7. Graf Stage 2

Kombinatorika:

Weight = 20

- ringan(5) → 15
  - ringan(5) → 10
    - ringan(5) → 5
      - ringan(5) → 0 ✓ [R,R,R,R]
    - sedang(10) → 0 ✓ [R,R,S]
  - sedang(10) → 5
    - (skip karena akan duplikat)
- sedang(10) → 10
  - sedang(10) → 0 ✓ [S,S]
- berat(15) → 5
  - ringan(5) → 0 ✓ [B,R]

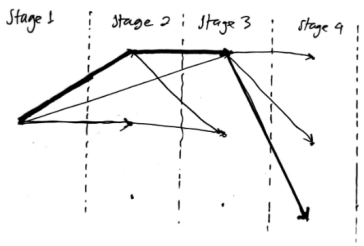
Total kombinasi unik: 4

3. Stage 3

```
Stage 3/7 (Total Weight: 25)
Total possible enemy combinations: 5

Encountered enemies in this stage:
5x ringan (Weight: 5)

Continue to next stage? (y/n): y

Available stages: 4a, 4b, 4c
Choose next stage: 4c
```

Gambar 8. Output Stage 3

Graf Sementara:



Gambar 9. Graf Stage 3

Kombinatorika:

Weight = 25

```
├── ringan(5) → 20
│   ├── ringan(5) → 15
│   │   ├── ringan(5) → 10
│   │   │   ├── ringan(5) → 5
│   │   │   │   └── ringan(5) → 0 ✓ [R,R,R,R,R]
│   │   │   └── sedang(10) → 0 ✓ [R,R,R,S]
│   │   └── berat(15) → 0 ✓ [R,R,B]
│   └── sedang(10) → 15
│       ├── sedang(10) → 5
│       │   └── ringan(5) → 0 ✓ [S,S,R]
│       └── berat(15) → 0 ✓ [S,B]
└── berat(15) → 10
    └── sedang(10) → 0 ✓ [B,S]
```

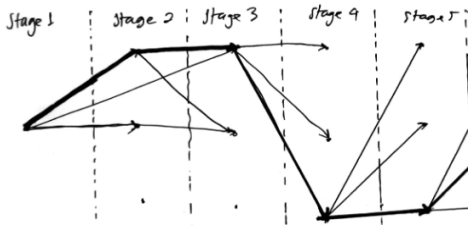
Total kombinasi unik: 6

4. Stage 4

```
Stage 4/7 (Total Weight: 30)
Total possible enemy combinations: 7

Encountered enemies in this stage:
1x berat (Weight: 15)
1x ringan (Weight: 5)
1x sedang (Weight: 10)

Continue to next stage? (y/n): y

Available stages: 5a, 5b, 5c
Choose next stage: 5c
```

Gambar 10. Output Stage 4

Graf Sementara:



Gambar 11. Graf Stage 4

Kombinatorika:

Weight = 30

```
├── ringan(5) → 25
│   ├── ringan(5) → 20
│   │   ├── ringan(5) → 15
│   │   │   ├── ringan(5) → 10
│   │   │   │   ├── ringan(5) → 5
│   │   │   │   │   └── ringan(5) → 0 ✓ [R,R,R,R,R,R]
│   │   │   │   └── sedang(10) → 0 ✓ [R,R,R,R,S]
│   │   │   └── berat(15) → 0 ✓ [R,R,R,B]
│   │   └── sedang(10) → 10
│   │       └── sedang(10) → 0 ✓ [R,R,S,S]
│   └── sedang(10) → 20
│       ├── sedang(10) → 10
│       │   └── sedang(10) → 0 ✓ [S,S,S]
│       └── berat(15) → 5
│           └── ringan(5) → 0 ✓ [S,B,R]
└── berat(15) → 15
    └── berat(15) → 0 ✓ [B,B]
```

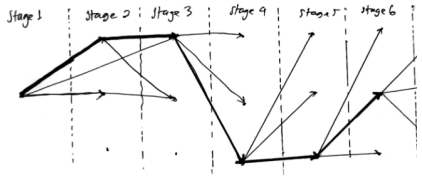
Total kombinasi unik: 7

5. Stage 5

```
Stage 5/7 (Total Weight: 35)
Total possible enemy combinations: 8

Encountered enemies in this stage:
5x ringan (Weight: 5)
1x sedang (Weight: 10)

Continue to next stage? (y/n): y

Available stages: 6a, 6b, 6c
Choose next stage: 6b
```

Gambar 12. Output Stage 5

Graf Sementara:



Gambar 13. Graf Stage 5

Kombinatorika:

Weight = 35

```
├── ringan(5) → 30
│   ├── ringan(5) → 25
│   │   ├── ringan(5) → 20
│   │   │   ├── ringan(5) → 15
│   │   │   │   ├── ringan(5) → 10
│   │   │   │   │   ├── ringan(5) → 5
│   │   │   │   │   │   └── ringan(5) → 0 ✓ [R,R,R,R,R,R,R]
│   │   │   │   │   └── sedang(10) → 0 ✓ [R,R,R,R,R,S]
│   │   │   │   └── berat(15) → 0 ✓ [R,R,R,R,B]
│   │   │   └── sedang(10) → 10
│   │   │       └── sedang(10) → 0 ✓ [R,R,R,S,S]
│   │   └── berat(15) → 5
│   │       └── ringan(5) → 0 ✓ [R,R,B,R]
│   └── sedang(10) → 25
│       ├── sedang(10) → 15
│       │   ├── sedang(10) → 5
│       │   │   └── ringan(5) → 0 ✓ [S,S,S,R]
```

└── berat(15) → 0 ✓ [S,S,B]
└── berat(15) → 20
├── berat(15) → 5
└── ringan(5) → 0 ✓ [B,B,R]
└── sedang(10) → 10
└── sedang(10) → 0 ✓ [B,S,S]

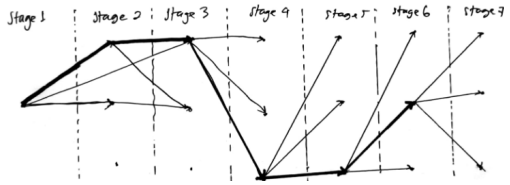
Total kombinasi unik: 9

6. Stage 6

```
Stage 6/7 (Total Weight: 40)
Total possible enemy combinations: 10

Encountered enemies in this stage:
2x ringan (Weight: 5)
3x sedang (Weight: 10)

Continue to next stage? (y/n): y

Available stages: 7a, 7b, 7c
Choose next stage: 7a
```

Gambar 14. Output Stage 6

Graf Sementara:



Gambar 15. Graf Stage 6

Kombinatorika:

Weight = 40

├── ringan(5) → 35 [sama seperti tree 35 + ringan]
├── sedang(10) → 30
├── sedang(10) → 20
├── sedang(10) → 10
└── sedang(10) → 0 ✓ [S,S,S,S]
└── berat(15) → 5
└── ringan(5) → 0 ✓ [S,S,B,R]
└── berat(15) → 25
├── berat(15) → 10
└── sedang(10) → 0 ✓ [B,B,S]
└── sedang(10) → 15
└── sedang(10) → 5
└── ringan(5) → 0 ✓ [B,S,S,R]

Total kombinasi unik: 11

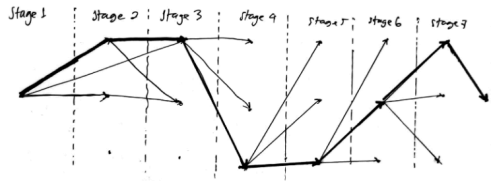
7. Stage 7

```
Stage 7/7 (Total Weight: 45)
Total possible enemy combinations: 12

Encountered enemies in this stage:
1x berat (Weight: 15)
2x ringan (Weight: 5)
2x sedang (Weight: 10)

BOSS FIGHT - Tartarus Boss
```

Gambar 16. Output Stage 7

notes: BOSS FIGHT terjadi setelah Stage 7 selesai

Graf Sementara:



Gambar 17. Graf Stage 7

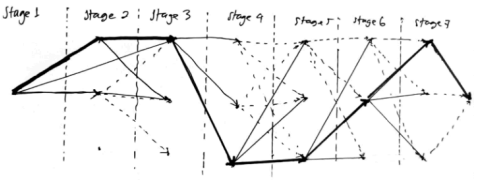
Kombinatorika:

Weight = 45

├── ringan(5) → 40 [sama seperti tree 40 + ringan]
├── sedang(10) → 35
├── sedang(10) → 25
├── sedang(10) → 15
├── sedang(10) → 5
└── ringan(5) → 0 ✓ [S,S,S,S,R]
└── berat(15) → 0 ✓ [S,S,S,B]
└── berat(15) → 30
├── berat(15) → 15
└── berat(15) → 0 ✓ [B,B,B]
└── sedang(10) → 20
├── sedang(10) → 10
└── sedang(10) → 0 ✓ [B,S,S,S]
└── berat(15) → 5
└── ringan(5) → 0 ✓ [B,B,R,S]

Total kombinasi unik: 13

Graf Akhir:



Gambar 18. Graf Akhir Chamber 1

CHAMBER 2:

Karena kasus yang terjadi mirip dengan Chamber 1, untuk Chamber 2 tidak ada analisis.

CHAMBER 3:

Pada Chamber 3, terdapat kasus yang berbeda, dimana terjadi loncatan stage:

1. Stage 1

```
Stage 1/7 (Total Weight: 15)
Total possible enemy combinations: 3

Encountered enemies in this stage:
3x ringan (Weight: 5)

Continue to next stage? (y/n): y

Available stages: 2a, 2b, 2c
Choose next stage: 2c
```

Gambar 19. Output Stage 1

2. Stage 2

```
Stage 2/7 (Total Weight: 20)
Total possible enemy combinations: 4

Encountered enemies in this stage:
4x ringan (Weight: 5)

Continue to next stage? (y/n): y

Available stages: 3a, 3b, 4a, 4b
Choose next stage: 4a
```

Gambar 20. Output Stage 2

3. Stage 3
   (Terjadi Loncatan Stage)

4. Stage 4

```
Stage 4/7 (Total Weight: 30)
Total possible enemy combinations: 7

Encountered enemies in this stage:
2x berat (Weight: 15)

Continue to next stage? (y/n): y

Available stages: 5a, 5b, 5c, 6a, 6b
Choose next stage: 6a
```

Gambar 21. Output Stage 4

5. Stage 5
   (Terjadi Loncatan Stage)

6. Stage 6

```
Stage 6/7 (Total Weight: 40)
Total possible enemy combinations: 10

Encountered enemies in this stage:
4x sedang (Weight: 10)

Continue to next stage? (y/n): y

Available stages: 7a, 7b
Choose next stage: 7a
```

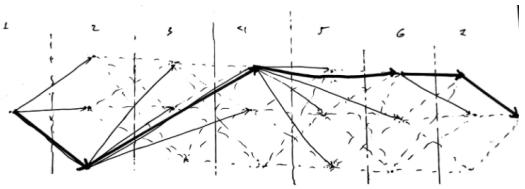
Gambar 22. Output Stage 6

7. Stage 7

```
Stage 7/7 (Total Weight: 45)
Total possible enemy combinations: 12

Encountered enemies in this stage:
1x berat (Weight: 15)
4x ringan (Weight: 5)
1x sedang (Weight: 10)

BOSS FIGHT - Elysium Boss
```

Gambar 23. Output Stage 7

Graf Akhir:



Gambar 24. Output Stage 8

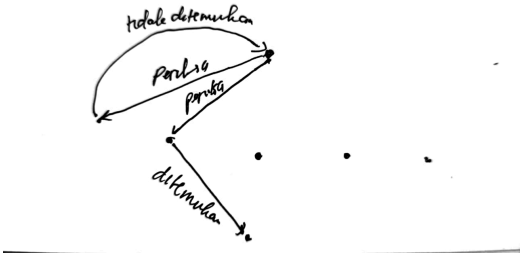
CHAMBER 4:

```
Entering Temple of Styx
Checking rooms for the key...
Checking room 1...
Checking room 2...
Key found in room 2!

Proceeding to Final Boss - Hades...

Final Boss Fight with Hades!
```

Gambar 25. Output Chamber 4

Karena terdapat 5 ruangan, maka persentase peluang lokasi kunci di masing-masing ruangan adalah 20%.

Graf untuk Chamber 4:



Gambar 26. Graf Chamber 4

V. Some Common Mistakes

1. Ketidakseimbangan Kombinasi Musuh

Bobot musuh yang tidak dirancang dengan baik dapat membuat permainan terasa terlalu mudah atau terlalu sulit. Sebagai contoh, jika bobot musuh di awal stage terlalu tinggi, pemain mungkin menghadapi tantangan yang tidak seimbang. Oleh karena itu, bobot musuh harus disesuaikan secara progresif sesuai dengan alur permainan, seperti peningkatan bertahap dari Tartarus hingga Elysium.

2. Kesalahan dalam Penerapan Peluang di Temple of Styx

Pada Temple of Styx, salah satu kesalahan umum adalah memberikan peluang yang tidak merata pada ruangan-ruangan untuk berisi kunci. Hal ini dapat menyebabkan pengalaman gameplay yang tidak adil bagi pemain. Solusinya adalah memastikan bahwa setiap ruangan memiliki probabilitas awal yang sama, dan peluang bersyarat diterapkan secara benar saat ruangan pertama tidak berisi kunci.

VI. Conclusion

Makalah ini membahas penerapan graf, kombinatorika, dan peluang dalam procedural generation untuk membangun peta dinamis di game Hades. Analisis menunjukkan bahwa pendekatan berbasis graf menciptakan struktur peta yang

fleksibel namun tetap terorganisasi, sementara kombinatorika digunakan untuk menghasilkan variasi musuh di setiap stage berdasarkan bobot total. Penambahan elemen peluang dalam Temple of Styx memberikan tantangan tambahan dengan mencari kunci menuju Final Boss, menciptakan gameplay yang tidak hanya dinamis tetapi juga penuh strategi.

Lebih jauh lagi, penelitian ini menunjukkan bahwa keragaman jalur, musuh, dan hadiah membuat procedural generation dalam Hades semakin menarik. Loncatan ke stage lebih jauh, meskipun meningkatkan kesulitan musuh, memberikan keuntungan strategis dengan mempercepat perjalanan menuju Final Boss. Dengan pendekatan ini, replayability dan variasi gameplay dapat ditingkatkan secara signifikan.

Ke depan, penelitian lebih lanjut dapat dilakukan untuk memperluas analisis terhadap mekanisme reward yang lebih kompleks, seperti menyesuaikan hadiah berdasarkan kesulitan musuh di setiap jalur. Selain itu, variasi musuh yang lebih beragam dapat diterapkan untuk meningkatkan tantangan strategis bagi pemain di setiap sesi permainan.

VII. Acknowledgment

Pertama, saya ingin mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas tuntunan-Nyalah saya dapat menyelesaikan masalah ini. Kedua, saya ingin mengucapkan terima kasih untuk seluruh dosen mata kuliah IF1220 Matematika Diskrit, khususnya untuk Arrival Dwi Sentosa, S.Kom., M.T. selaku dosen K-03 Jatinangor yang sudah menuntun saya menjelajahi dunia matematika diskrit. Dan saya juga berterima kasih untuk semua pihak lain yang secara tidak langsung sudah membantu saya dalam menyelesaikan makalah ini.

References

[1] Munir, R. (2024). Graf (Bagian 1). Diakses pada 6 Januari 2025, dari: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

[2] Munir, R. (2024). Graf (Bagian 2). Diakses pada 6 Januari 2025, dari: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf

[3] Munir, R. (2024). Graf (Bagian 3). Diakses pada 6 Januari 2025, dari: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf

[4] Munir, R. (2024). Kombinatorika (Bagian 1). Diakses pada 6 Januari 2025, dari: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/18-Kombinatorika-Bagian1-2024.pdf

[5] Munir, R. (2024). Kombinatorika (Bagian 2). Diakses pada 6 Januari 2025, dari: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/19-Kombinatorika-Bagian2-2024.pdf

[6] "Chambers and Encounters." Hades Wiki. Diakses pada 7 Januari 2025, dari: https://hades.fandom.com/wiki/Chambers_and_Encounters.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Jatinangor, 8 Januari 2025

Ivant Samuel Silaban