Maliki Interdisciplinary Journal (MIJ) eISSN: 3024-8140
Volume 2, Issue 8, 2024 pp. 55-62 http://urj.uin-malang.ac.id/index.php/mij/index

Penerapan Algoritma Recursive DFS dalam pembuatan Labirin Dinamis pada Game "Lingkaran Setan" dengan teknik Procedural Maze Generation

Afiifah Zain Raidah[1]*, Suci Wulandari[2], Rahmat Fauzan[3], Nazzil Akfa Said F[4]
[1, 2, 3, 4] *Program Studi Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim Malang*
*e-mail: *210605110150@student.uin-malang.ac.id*

Kata Kunci:
Recursive DFS; labirin dinamis; game Lingkaran Setan; Procedural Maze Generation

Keywords:
Recursive DFS; labyrinth; *game Lingkaran Setan*; Procedural Maze Creationafter

A B S T R A K

Penelitian ini mengeksplorasi penerapan teknik Procedural Maze Generation dalam menciptakan labirin dinamis pada permainan "Lingkaran Setan". Labirin dinamis diperlukan untuk memberikan pengalaman bermain yang menarik dan menantang, memastikan variasi yang konstan dalam setiap sesi permainan. Kami menggunakan algoritma Procedural Maze Generation untuk secara otomatis menciptakan labirin dengan struktur yang kompleks namun dapat diakses, memungkinkan pemain untuk menjelajahi lingkungan permainan dengan keunikan setiap kali permainan dimulai. Penelitian ini menitikberatkan pada integrasi algoritma Procedural Maze Generation dengan Game Lingkaran Setan, menghasilkan lingkungan yang dinamis dan selalu menantang. Hasil eksperimen menunjukkan bahwa penggunaan teknik ini dapat meningkatkan tingkat replayability dan kepuasan pemain. Selain itu, kami mengukur performa algoritma dalam hal efisiensi dan kecepatan pembangunan labirin untuk memastikan pengalaman bermain yang mulus. Temuan dari penelitian ini memberikan kontribusi pada pemahaman lebih lanjut tentang penerapan teknik Procedural Maze Generation dalam konteks permainan video, memberikan dasar untuk pengembangan game yang lebih dinamis dan menarik di masa depan.

A B S T R A C T

This research explores the application of Procedural Maze Generation techniques in creating dynamic mazes for the game "Lingkaran Setan". Dynamic mazes are necessary to provide an engaging and challenging gameplay experience, ensuring constant variation in each game session. We use a Procedural Maze Generation algorithm to automatically create mazes with complex yet accessible structures, allowing players to explore the game environment uniquely each time the game starts. This research focuses on the integration of the Procedural Maze Generation algorithm with the game "Lingkaran Setan," resulting in dynamic and always challenging environments. Experimental results show that using this technique can enhance replayability and player satisfaction. Additionally, we measure the algorithm's performance in terms of efficiency and maze-building speed to ensure a smooth gaming experience. The findings from this study contribute to a further understanding of the application of Procedural Maze Generation techniques in the context of video games, providing a foundation for developing more dynamic and engaging games in the future.

Pendahuluan

Indonesia merupakan salah satu negara dengan jumlah penggemar game yang sangat tinggi. Di sekitar kita, terlihat bahwa mayoritas orang memainkan game, baik untuk hiburan sesekali maupun sebagai rutinitas harian. Mulai dari anak-anak hingga orang dewasa, banyak yang menikmati permainan ini. Oleh karena itu, industri game menawarkan peluang bisnis yang sangat menjanjikan bagi para pengembang game. Besarnya minat terhadap game menciptakan peluang besar bagi profesional di bidang ini. Bahkan, hal ini juga membuka kesempatan bagi pemula yang ingin belajar dan serius dalam pembuatan game (Fadila et al., 2023).

Dalam era teknologi dan informasi yang melanda saat ini, penggunaan teknologi semakin merasuk ke berbagai aspek kehidupan dengan tingkat intensitas yang semakin tinggi. Bukan hanya kalangan dewasa, tetapi mulai dari anak-anak hingga orang dewasa, banyak yang menyisihkan waktu mereka untuk terlibat dalam berbagai jenis permainan berbasis komputer. Jenis permainan ini meliputi berbagai tingkat kompleksitas, mulai dari yang sederhana hingga yang memiliki aturan permainan yang lebih rumit .

Salah satu bentuk permainan yang semakin diminati adalah permainan labirin. Labirin, sebagai jenis permainan yang menantang, menjadi daya tarik tersendiri di tengah kepopuleran permainan berbasis teknologi. Labirin, menurut definisi, adalah sebuah teka-teki yang menantang pemain untuk menemukan jalan keluar. Selama perjalanan melalui labirin, pemain akan menghadapi berbagai rintangan atau halangan yang harus diatasi untuk mencapai tujuan akhir (Adiguna & Swanjaya, 2020). Untuk menciptakan game labirin dengan arena yang dinamis, diperlukan sebuah pembangkit labirin atau maze generator. Maze generator ini bertujuan untuk menghasilkan labirin yang berbeda setiap kali pemain memainkan game tersebut (Krisdiawan et al., 2022). Dalam konteks ini, pemain diundang untuk menjelajahi jalur-jalur yang kompleks, memecahkan teka-teki, dan mencapai tujuan akhir dalam labirin yang dihadapkan. Labirin, sebagai bentuk puzzle yang telah lama menjadi bagian integral dari berbagai tantangan intelektual, terus menghadirkan daya tariknya dalam dunia gaming. Dengan sejumlah jalur yang kompleks dan bercabang-cabang, labirin menantang pemain untuk memecahkan teka-teki dengan menemukan jalur yang benar, mulai dari titik awal hingga titik akhir . Labirin pada umumnya dibuat untuk tujuan hiburan. Dalam kehidupan nyata, labirin sering dibuat di taman atau ruangan dengan pembatas berupa pagar atau tembok, dengan ukuran yang bervariasi (Rahmasuci et al., 2019).

Dalam dunia game, pengalaman menjelajahi labirin semakin menarik ketika mengintegrasikan algoritma DFS (Depth-First Search) atau Recursive DFS dalam proses pembuatannya. Algoritma DFS membuka dimensi baru dalam pembuatan labirin, memungkinkan pengembang game untuk menciptakan pengalaman bermain yang tak terduga dan dinamis. Dengan menggabungkan Recursive DFS dalam kerangka teknik procedural maze generation, "Lingkaran Setan" menjadi wahana eksplorasi yang menarik, di mana algoritma ini memberikan kontribusi signifikan dalam menciptakan labirin yang unik setiap kali permainan dimulai.

Recursive Depth-First Search (DFS) adalah implementasi khusus dari algoritma DFS yang menggunakan teknik rekursi untuk menjelajahi struktur data, seperti graf atau

pohon. Menurut (Needham & Hodler, n.d.), DFS (Depth-First Search) adalah algoritma traversal graf yang mendasar. Algoritma ini dimulai dengan memilih sebuah simpul, kemudian memilih salah satu simpul tetangganya, dan berjalan sejauh mungkin sebelum melakukan backtracking. Algoritma ini secara mendalam mengeksplorasi suatu cabang hingga mencapai titik akhir sebelum kembali ke cabang sebelumnya dan melanjutkan pencarian. Algoritma Recursive Depth first search (DFS) ini telah menjadi salah satu pendekatan yang signifikan, terutama pada game berbasis labirin (*Jurnal Digit Vol1 No2*, 2024).

Dalam artikel ini, kita akan membahas secara rinci bagaimana penerapan Algoritma Recursive DFS pada "Lingkaran Setan" mempengaruhi pembuatan labirin dinamis, memberikan pengalaman bermain yang menghibur dan penuh tantangan. Game "Lingkaran Setan" menghadirkan tantangan bertahan hidup dari labirin yang penuh godaan setan. Pemain dalam game ini harus mengumpulkan item tasbih untuk menunda pergerakan musuh agar bisa mencari jalan keluar dari labirin tersebut. Pembuatan labirin pada game ini menggunakan bantuan AI Maze Generator dengan menerapkan metode DFS (Depth First Search).

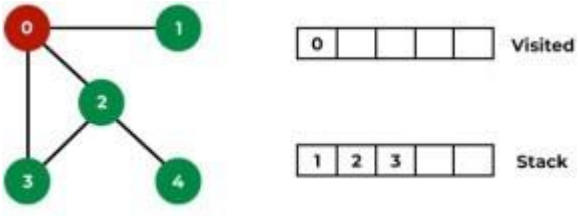
Metode

Algoritma Recursive DFS ini memulai penelusuran dari suatu simpul tertentu dan mengeksplorasi sejauh mungkin di setiap cabang sebelum melakukan backtracking. Konsep dasarnya adalah memulai dari suatu simpul tertentu, kemudian menjelajahi sejauh mungkin di setiap cabang sebelum melakukan backtracking. DFS bekerja dengan cara mempertahankan sebuah tumpukan (atau menggunakan rekursi) untuk melacak simpul-simpul yang akan dikunjungi. Penerapan algoritma DFS dengan menggunakan rekursi dapat dilihat pada ilustrasi berikut:

1. Terdapat Graph dan table visited dan stack yang kosong

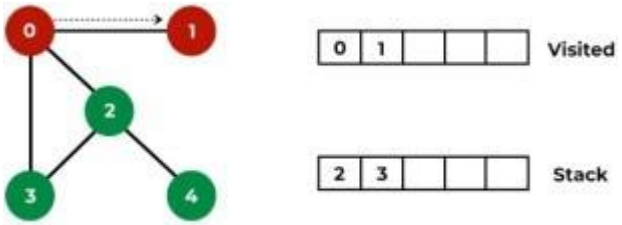
Gambar 1. Tahap pertama DFS

2. Telusuri node 0 dan memasukkan tetangga node 0 kedalam stack



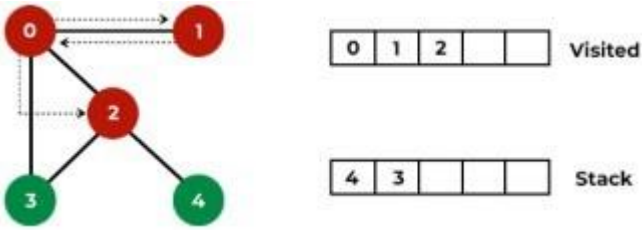
Gambar 2. Tahap Kedua DFS

3. Telusuri node 1



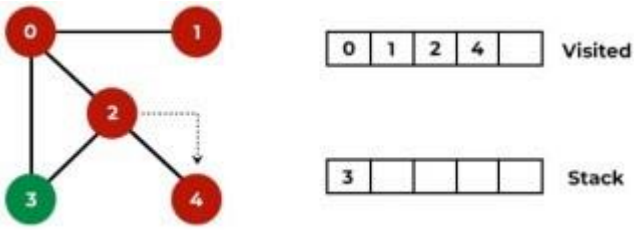
Gambar 3. Tahap Ketiga DFS

4. Ganti penelusuran, karena node 1 adalah node yang paling ujung pada graph tersebut, maka penelusuran akan kembali ke node 0, dan dilanjutkan dengan penelusuran node 2. Node 4 ditambahkan pada stack karena merupakan tetangga 2.



Gambar 4. Tahap keempat DFS

5. Penelusuran menuju node terdalam, yaitu node 4



Gambar 5. Tahap kelima DFS

6. Kembali ke node 2, karena node 4 adalah node yang paling ujung, maka penelusuran akan kembali ke node 2 dan dilanjutkan ke node 3.
7. Penelusuran berakhir, karena semua node sudah dikunjungi maka penelusuran selesai.

Pembahasan

Penerapan algoritma DFS pada labirin game 3D bisa dilihat dari kode program berikut serta penjelasannya

```
void Generate(int x,int z) {
  if (CountSquareNeighbours(x, z) >= 2)
  return; map[x, z] = 0;
  directions.shuffle();
  Generate(x + directions[0].x, z+ directions[0].z);
  Generate(x + directions[1].x, z + directions[1].z);
  Generate(x + directions[2].x, z + directions[2].z);
  Generate(x + directions[3].x, z + directions[3].z);
}
```

Kode diatas akan menghasilkan labirin sesuai dengan alur kerja Algoritma DFS yang sudah dijelaskan sebelumnya. berikut adalah penjelasan dari kode:

1. Fungsi Generate

Fungsi Generate menerima dua parameter, x dan z, yang menggambarkan koordinat posisi saat ini dalam matriks labirin (map). Koordinat (x, z) merepresentasikan lokasi atau sel yang sedang diproses.

2. Pengecekan Tetangga

Fungsi CountSquareNeighbours(x, z) digunakan untuk menghitung jumlah tetangga sel pada posisi (x, z) yang sudah diisi (dengan nilai 1). Jika jumlah tetangga yang sudah diisi lebih dari atau sama dengan 2, maka fungsi Generate langsung keluar tanpa melakukan lebih lanjut. Hal ini mencegah agar tidak terbentuk cabang-cabang kecil yang dapat menghasilkan labirin yang terlalu kompleks.

3. Penandaan sel (map[x, z] = 0):

Jika kondisi pada langkah sebelumnya tidak terpenuhi, nilai sel pada posisi (x, z) dalam matriks map diubah menjadi 0. Ini menandakan bahwa jalur atau ruang kosong sedang dibuat pada posisi ini.

4. Pengacakan Arah (directions.shuffle()):

Dalam langkah ini, terdapat objek directions yang mungkin berisi empat vektor arah yang mewakili perpindahan ke tetangga atas, bawah, kiri, dan kanan. Dengan memanggil fungsi shuffle(), urutan arah ini diacak untuk memilih urutan yang berbeda setiap kali fungsi Generate dipanggil. Hal ini memberikan variasi pada arah pergerakan dan menghasilkan labirin yang lebih bervariasi.

5. Pemanggilan Rekursif

Pada bagian ini, terdapat empat pemanggilan rekursif untuk setiap arah yang telah diacak. Dengan menggunakan indeks 0 hingga 3 dari objek directions, fungsi Generate dipanggil kembali untuk menjelajahi tetangga sel saat ini. Setiap pemanggilan rekursif menyebabkan proses serupa dilakukan pada sel tetangga, dan proses ini terus berlanjut hingga kondisi berhenti terpenuhi atau seluruh labirin terbentuk.

Langkah selanjutnya melibatkan penerapan langkah-langkah tersebut dalam skrip untuk menciptakan labirin. Dalam pelaksanaan ini, dua model kubus digunakan sebagai representasi 3D untuk labirin, sesuai dengan Gambar 1, dan Gambar 2.

Selanjutnya dalam pembentukan labirin ini digunakan juga fungsi shuffle yang nantinya akan mengantre labirin dengan jumlah dan letak yang bervariasi. Berikut kode yang digunakan untuk mengimplementasikan fungsi shuffle dalam pembuatan labirin.

```
public static void shuffle<T>(this IList<T> list)
 {
   int       n       =
   list.Count;
   while (n > 1)
   {
     n--;
     int k = rng.Next(n + 1);
     T Value = list[k];
     list[k] = list[n]; list[n] = Value;
```

Berikut penjelasan kode

1. Deklarasi Metode Ekstensi shuffle

Metode ini dideklarasikan sebagai metode ekstensi untuk objek yang mengimplementasikan antarmuka IList<T>. Dengan kata lain, metode ini dapat digunakan pada tipe data yang merupakan daftar (list).

2. Inisialisasi Jumlah Elemen

Variabel n diinisialisasi dengan jumlah total elemen dalam daftar (list). Ini adalah langkah awal untuk menentukan seberapa banyak elemen yang akan diacak.

3. Looping untuk Pengacakan

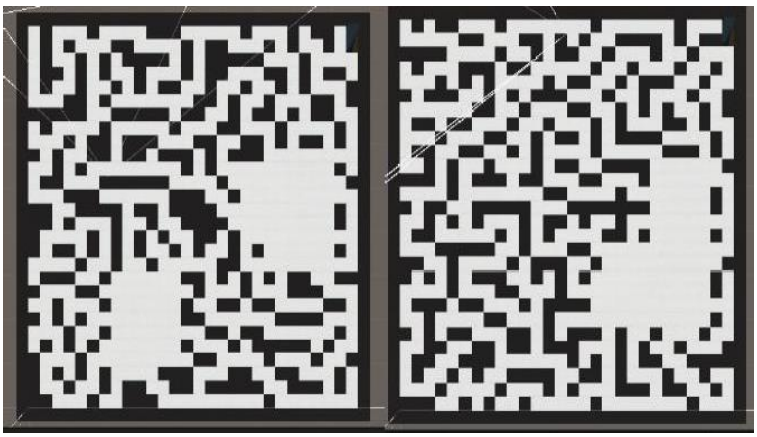
Dalam setiap iterasi loop, nilai n dikurangi satu (n--) untuk menentukan rentang indeks elemen yang dapat diakses. Selanjutnya, nilai indeks acak k dihasilkan dengan menggunakan metode Next dari objek rng. Nilai k akan berada dalam rentang antara 0 dan n, inklusif.

4. Penukaran Elemen

Nilai elemen pada indeks ‘k’ ditukar dengan nilai elemen pada indeks ‘n’. Proses pertukaran elemen ini merupakan inti dari algoritma Fisher-Yates yang menghasilkan pengacakan. Proses ini diulang hingga hanya ada satu elemen yang belum diacak.

Dalam konteks labirin, metode shuffle digunakan pada list directions(pada kode generate) yang berisi empat vektor arah. Penerapan shuffle pada directions menyebabkan urutan vektor arah ini diacak setiap kali fungsi Generate dipanggil.

Dengan mengacak arah perpindahan, proses pembentukan labirin menjadi lebih dinamis dan bervariasi, menciptakan jalur yang berbeda-beda setiap kali labirin dibangun. Hal ini memberikan keunikan dan tantangan pada pengalaman bermain labirin dalam game.



Gambar 6. Hasil generate labirin ke 1 dan ke 2

Ketika program dieksekusi, labirin yang dihasilkan menampilkan variasi bentuk yang bergantung pada titik awal pembentukannya. Hal ini dapat diamati melalui perbandingan antara Gambar 1 dan Gambar 2, yang menunjukkan perbedaan hasil pembentukan labirin. Proses pembentukan labirin dimulai dari suatu titik awal tertentu, seperti yang terlihat pada Gambar 1 dan Gambar 2, menghasilkan bentuk dan struktur labirin yang beragam. Dengan menggunakan pendekatan algoritma DFS dan konsep rekursi, program mampu menciptakan labirin yang dinamis dan bervariasi setiap kali dijalankan, memberikan pengalaman bermain yang unik pada setiap iterasi permainan.

Kesimpulan dan Saran

Implementasi Algoritma Recursive DFS pada pembuatan labirin dalam game "Lingkaran Setan" sukses menciptakan labirin yang dinamis, kompleks, dan bervariasi. Keberhasilan algoritma dalam penjelajahan rekursif dan pembentukan labirin memberikan daya tarik dan tingkat tantangan yang optimal dalam pengalaman bermain. Dengan demikian, Algoritma DFS terbukti menjadi pilihan yang efektif untuk menghasilkan labirin menarik dalam konteks pengembangan game.

Daftar Pustaka

Adiguna, Y., & Swanjaya, D. (2020). Implementasi algoritma backtracking untuk mencari jalan keluar labirin.

Fadila, J. N., Nugroho, F., Artanti, V., Rohma, S. A., Huda, M. K., & Priambudi, N. S. (2023). Penerapan Hfsm pada Game 3d “Petualang Qur’an.” Jurnal Ilmiah Informatika, 11(01), 15–21. https://doi.org/10.33884/jif.v11i01.6538

Krisdiawan, R. A., Fitriani, A., & Budianto, H. (2022). Penerapan Algoritma Recursive Backtracking sebagai Maze Generator pada Game Labirin Aksara Sunda. Media Jurnal Informatika, 14(1), 31. https://doi.org/10.35194/mji.v14i1.2326

Needham, M., & Hodler, A. E. (n.d.). Graph Algorithms.

Putri, N. M., Umri, I. R., Azmi, M. I. N., Amriadi, A., Shaffira, F., Karami, A. F., & Nugroho, F. (2024). Implementasi Algoritma Recursive Depth First Search pada Game Labirin 3d berbasis desktop. Jurnal Digit, 14(1), 1-8.

Rahmasuci, M., S, H. H., Azizah, M., Wulandari, P., A, D. A., & Bukhori, S. (2019). Strategi menemukan jalan keluar labirin dengan waktu tercepat menggunakan metode DFS. INFORMAL: Informatics Journal, 3(1), 12. https://doi.org/10.19184/isj.v3i1.9852