IT-EXPLORE
Jurnal Penerapan Teknologi Informasi danc Komunikasi
ISSN 2828-7940, e-ISSN 2829-1727, DOI: 10.24246/ v4i3.2025.pp352-364
Volume 04 Nomor 03 Tahun 2025
Journal homepage: https://ejournal.uksw.edu/itexplore

Implementasi algoritma *random walk* untuk menghasilkan procedural map pada game *scrollbearer*

Yosafat Rendi Prayoga[1], Sri Winarso Martyas Edi, [2]
[1,2] Satya Wacana Christian University, Salatiga City, Central Java, Indonesia

Informasi Artikel

Riwayat Artikel:

Dikirim:
21 April 2025
Direvisi:
19 Mei 2025
Diterima:
1 Juni 2025

Kata Kunci:

random walk,
procedural generation,
Unity,
game roguelike,
replayability

Keywords:

random walk,
procedural generation,
Unity,
roguelike game,
replayability

This is an open access article under the CC BY-SA license

ABSTRAK

Perkembangan industri game yang pesat menuntut inovasi dalam desain peta permainan untuk menjaga pengalaman bermain yang dinamis dan tidak repetitif. Penelitian ini bertujuan mengimplementasikan algoritma *Random Walk* guna menghasilkan peta prosedural pada game *Scrollbearer*, sebuah game *roguelike* berbasis C# di *Unity*. Algoritma ini bekerja dengan membentuk jalur koridor dan ruangan secara acak dari titik awal, dilengkapi mekanisme penanganan *dead ends* dan integrasi *Corridor First* untuk memastikan konektivitas peta. Metode penelitian menggunakan pendekatan *Research and Development* (R&D) dengan evaluasi kuantitatif melalui kuesioner Skala *Likert*. Hasil uji coba menunjukkan bahwa peta prosedural berhasil meningkatkan *replayability*, dengan 79,5% pemain lebih memilih mode ini karena variasi layout yang unik dan tantangan lebih tinggi (skor 4,07/5). Kepuasan eksplorasi pada mode prosedural juga lebih baik (skor 4,34/5) dibandingkan mode statis. Temuan ini membuktikan efektivitas algoritma *Random Walk* dalam menciptakan peta dinamis untuk genre *roguelike*, sekaligus menjadi dasar pengembangan fitur tambahan seperti variasi *biome* dan *random events*.

ABSTRACT

The rapid growth of the gaming industry demands innovation in map design to maintain dynamic and non-repetitive gameplay experiences. This study aims to implement the Random Walk algorithm to generate procedural maps for Scrollbearer, a roguelike game developed using C# in Unity. The algorithm constructs random corridors and rooms from a starting point, integrated with dead end handling and Corridor First mechanics to ensure map connectivity. The research applies a Research and Development (R&D) approach with quantitative evaluation through Likert-scale questionnaires. Test results demonstrate that procedural maps significantly enhance replayability, with 79.5% of players preferring this mode due to unique layout variations and higher challenge levels (score 4.07/5). Exploration satisfaction in the procedural mode also outperformed the static mode (score 4.34/5). These findings validate the effectiveness of the Random Walk algorithm in creating dynamic maps for roguelike games, while providing a foundation for future enhancements such as biome diversity and random events.

Corresponding Author:

Yosafat Rendi Prayoga
Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana
Jl. Diponegoro No.52-60, Sidorejo, Salatiga, Jawa Tengah, Indonesia
Email: 672021055@student.uksw.edu

1. PENDAHULUAN

Industri game terus berkembang dengan pesat, mendorong inovasi dalam pengembangan teknologi dan metode pembuatan konten permainan. Salah satu elemen penting dalam pengembangan game adalah desain level atau peta permainan, yang berfungsi untuk menciptakan pengalaman bermain yang menarik, menantang, dan bervariasi [1]. Dalam genre tertentu, seperti *roguelike*, peta yang dinamis menjadi aspek krusial yang mempengaruhi daya tarik permainan. Oleh karena itu, pendekatan *procedural generation* semakin banyak diterapkan untuk menciptakan level yang unik dan tidak repetitif setiap kali permainan dimulai [2].

Procedural generation adalah metode untuk menghasilkan konten secara otomatis menggunakan algoritma, sehingga pengembang tidak perlu merancang setiap level secara manual [3]. Salah satu algoritma yang umum digunakan dalam *procedural map generation* adalah *Random Walk Algorithm*. Algoritma ini bekerja dengan cara memulai dari satu titik awal dan bergerak secara acak dalam berbagai arah, membentuk jalur atau area yang kemudian digunakan sebagai dasar pembuatan peta permainan [4].

Dalam konteks pengembangan game, algoritma *Random Walk* sering diterapkan dalam pembuatan peta permainan, terutama dalam genre roguelike yang membutuhkan level yang selalu berubah di setiap sesi permainan. Salah satu varian dari algoritma ini yang banyak digunakan adalah *Drunkard's Walk*, yang mensimulasikan gerakan acak seperti orang mabuk yang berjalan tanpa pola tertentu [5]. Algoritma ini memungkinkan pembuatan peta yang tidak dapat diprediksi oleh pemain, sehingga meningkatkan *replayability* dan tantangan dalam permainan [6].

Salah satu game engine yang banyak digunakan dalam pengembangan game berbasis *procedural generation* adalah *Unity*. *Unity* menyediakan fleksibilitas dalam implementasi berbagai algoritma dengan menggunakan bahasa pemrograman C#, yang memiliki performa tinggi dan banyak digunakan dalam industri game [7]. Dengan fitur-fitur yang mendukung *procedural generation*, Unity memungkinkan pengembang untuk mengimplementasikan algoritma seperti *Random Walk* secara efisien.

Penelitian ini bertujuan untuk mengimplementasikan *Random Walk Algorithm* dalam menghasilkan prosedural *map* untuk game *Scrollbearer*, sebuah game bergenre roguelike yang dikembangkan oleh penulis. Dalam game ini, variasi peta yang dinamis menjadi elemen utama dalam mempertahankan ketertarikan pemain. Dengan menerapkan algoritma *Random Walk* menggunakan bahasa C# di *Unity*, diharapkan game ini dapat menghadirkan level yang selalu unik dan tidak repetitif, sehingga memberikan pengalaman bermain yang menarik dan penuh tantangan bagi pemain.

Rumusan masalah dalam penelitian ini adalah bagaimana algoritma *Random Walk* dapat diimplementasikan secara efektif untuk menghasilkan *procedural map* yang dinamis dan variatif pada game *Scrollbearer*, serta apakah penggunaan algoritma tersebut dapat meningkatkan kualitas pengalaman bermain dan *replayability* dari game tersebut.

Tujuan dari penelitian ini adalah untuk mengimplementasikan algoritma *Random Walk* guna menghasilkan prosedural map pada game *Scrollbearer*, serta mengevaluasi efektivitas algoritma tersebut dalam menciptakan peta yang variatif dan meningkatkan kualitas pengalaman bermain serta *replayability*.

Batasan dalam penelitian ini adalah fokus hanya pada eksplorasi dan implementasi algoritma *Random Walk* dalam *Unity* dengan menggunakan bahasa pemrograman C#. Evaluasi difokuskan pada pengaruh algoritma tersebut terhadap kualitas pengalaman bermain dan *replayability* game *Scrollbearer*. Penelitian ini tidak mencakup pengembangan jangka panjang atau pembaruan algoritma setelah periode penelitian berakhir.

2. KAJIAN PUSTAKA

2.1. Penelitian Terdahulu

Penelitian tentang identifikasi node esensial dalam network pharmacology berbasis jaringan multilayer dengan algoritma *random walk* telah dilakukan sebelumnya [8]. Penelitian ini bertujuan untuk mengaplikasikan jaringan multilayer dalam *network pharmacology* untuk menganalisis mekanisme aksi obat dari perspektif keseluruhan. Hasilnya menunjukkan bahwa kombinasi algoritma *random walk* dengan jaringan *multilayer* dapat mengukur pentingnya *node* dari seluruh jaringan, bukan hanya dari satu lapisan. Sebuah jaringan empat lapis dibangun berdasarkan data proses aksi resep, yang terdiri dari bahan, protein target, jalur metabolisme, dan penyakit. Algoritma *random walk* digunakan untuk menghitung *betweenness centrality* dari *node* lapisan protein untuk mendapatkan peringkat kepentingannya. Dengan metode ini, 10% protein teratas yang memainkan peran kunci dalam pengobatan berhasil diidentifikasi. Hasil penelitian menunjukkan bahwa akurasi metode ini setara dengan metode topologi jaringan lapis tunggal. Penelitian ini membuktikan bahwa jaringan *multilayer* yang dikombinasikan dengan algoritma *random walk* adalah metode efektif untuk pra-penyaringan protein target vital terkait resep.

Penelitian lain mengembangkan algoritma *random walk* dengan pendekatan preferensi untuk prediksi tautan dalam jaringan kompleks [9]. Penelitian ini bertujuan untuk mengembangkan metode efisien yang meningkatkan algoritma *random walk* lokal dengan mendorong *random walk* untuk bergerak, pada setiap langkah, menuju *node* yang memiliki pengaruh lebih kuat. Hasilnya menunjukkan bahwa metode ini memiliki akurasi prediksi yang lebih tinggi dibandingkan dengan pendekatan prediksi link lainnya yang berbasis kesamaan, baik metode lokal, kuasi-lokal, maupun global. Metode ini diuji pada 11 jaringan dunia nyata dan terbukti lebih unggul dalam memprediksi tautan masa depan, yang terhapus, dan tautan redundan menggunakan tautan yang ada dalam grafik.

Penelitian lain menerapkan algoritma *random walk* dalam konteks pengembangan game dua dimensi yang bertujuan mengatasi masalah repetisi dalam pembuatan map pada game *The Last Hope* [10]. Penelitian ini menerapkan metode *Random Walk* untuk menghasilkan map secara prosedural. Hasilnya menunjukkan bahwa teknik ini mampu menciptakan variasi map yang dinamis di setiap sesi permainan, sehingga mengurangi rasa bosan yang biasanya dialami pemain. Algoritma *Random Walk* diimplementasikan menggunakan bahasa pemrograman C# di *Unity* 2D, dan terbukti meningkatkan aspek *replayability* dalam game. Dengan adanya variasi map yang terus berubah, pengalaman bermain menjadi lebih menarik dan tidak monoton. Penelitian ini berimplikasi pada peningkatan kualitas pengalaman bermain, khususnya pada game *The Last Hope* maupun game lain bergenre *roguelike*.

Berbeda dari penelitian-penelitian sebelumnya, penelitian ini secara khusus mengimplementasikan algoritma *random walk* pada game *Scrollbearer*, yang merupakan game orisinal buatan penulis sendiri dengan genre *roguelike*. Penelitian ini tidak hanya fokus pada penciptaan variasi map secara prosedural, tetapi juga menguji bagaimana penerapan algoritma random walk berdampak langsung terhadap pengalaman bermain dan *replayability* dalam game. Ciri khas dari penelitian ini terletak pada integrasi algoritma dengan sistem *gameplay* aktif dan evaluasi efeknya terhadap keterlibatan pemain, bukan sekadar pengujian teknis atau struktural algoritma semata.

2.2. Landasan Teori

Procedural generation adalah penggunaan algoritma dan aturan untuk secara otomatis membuat konten seperti dunia virtual, jaringan jalan raya, dan elemen permainan, bukan dengan tangan. Pendekatan ini memungkinkan terciptanya konten yang beragam dan realistis dengan mengikuti serangkaian parameter dan batasan yang telah ditentukan sebelumnya [11].

Roguelike merupakan *genre* permainan yang ditandai dengan level yang dibuat secara prosedural dan fitur *permadeath*, di mana pemain harus mengulang permainan dari awal ketika karakter yang dimainkan mati [12].

Random walk adalah suatu proses statistik yang menghasilkan pergerakan acak dari suatu titik awal ke titik lain dalam suatu ruang. Proses ini dilakukan dengan cara menghitung perubahan posisi dalam setiap

langkah yang diambil, dengan probabilitas yang sama untuk bergerak ke arah mana pun dalam ruang tersebut [13].

Replayability dalam game mengacu pada kemampuan game untuk mempertahankan daya tarik dan ketertarikannya selama beberapa kali permainan, mendorong pemain untuk terus bermain dan menjelajahi game tanpa merasa bosan atau jenuh. Konsep ini sangat penting dalam pengembangan game, karena dapat secara signifikan mempengaruhi umur panjang game dan kepuasan pemain secara keseluruhan [14].

Corridor first adalah metode dalam *procedural map generation* yang berfokus pada pembangunan koridor terlebih dahulu sebelum menempatkan ruangan atau elemen lain dalam peta permainan [1]. Teknik ini sering digunakan dalam *dungeon generation*, terutama dalam game *roguelike*, untuk memastikan bahwa semua bagian peta saling terhubung, sehingga menghindari area yang terisolasi [2].

Perlin noise adalah algoritma berbasis gradien noise yang dikembangkan oleh Ken Perlin pada tahun 1983 untuk menghasilkan tekstur dengan transisi yang lebih halus dan alami dalam grafik komputer [15]. Algoritma ini digunakan secara luas dalam *terrain generation*, *procedural texture mapping*, dan berbagai aplikasi lainnya dalam game development.

Skala *Likert* merupakan teknik pengukuran yang diperkenalkan oleh Rensis Likert pada tahun 1932, digunakan untuk mengevaluasi sikap atau pendapat responden melalui sejumlah pernyataan dengan pilihan tingkat persetujuan mulai dari "sangat tidak setuju" hingga "sangat setuju". [16]. Skala ini banyak digunakan dalam penelitian sosial dan kuantitatif untuk mengukur persepsi dan sikap manusia terhadap suatu fenomena [17].

NPC (*Non-Player Character*) adalah karakter dalam permainan video yang dikendalikan oleh kecerdasan buatan (AI) atau skrip komputer, bukan oleh pemain manusia [18]. NPC digunakan dalam berbagai peran, seperti karakter pendukung, musuh, atau elemen cerita yang membantu menghidupkan dunia game dan memberikan pengalaman bermain yang lebih dinamis [18].

C# merupakan bahasa pemrograman *high-level* yang dikembangkan oleh Microsoft sebagai bagian dari platform .NET. Bahasa ini dirancang sebagai bahasa pemrograman yang berorientasi objek dan sering digunakan dalam pengembangan aplikasi *desktop*, *web*, dan *game*. Dalam pengembangan *game*, C# menjadi bahasa utama yang digunakan dalam Unity untuk mengontrol *game logic*, *physics*, AI, dan *rendering* [19].

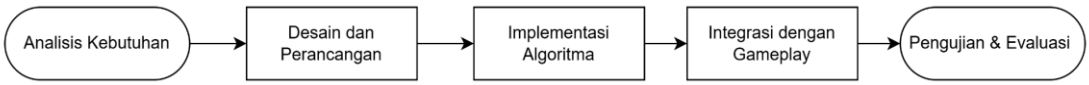
Unity adalah game engine lintas platform yang dikembangkan oleh Unity Technologies, yang memungkinkan pengembang untuk membuat game 2D, 3D, serta pengalaman berbasis realitas virtual dan *augmented reality*. Unity memiliki lingkungan pengembangan terpadu yang mendukung berbagai fitur, seperti *physics engine*, *rendering engine*, *audio processing*, dan integrasi dengan berbagai *platform*, termasuk PC, konsol, dan perangkat seluler [7].

3. METODE PENELITIAN

3.1. Desain Penelitian

Penelitian ini menggunakan metode *Research and Development (R&D)* dengan pendekatan kuantitatif. Proses penelitian ini diawali dengan pengembangan game dasar yang mencakup implementasi mekanika gameplay agar permainan dapat dijalankan. Selanjutnya, dilakukan pengembangan algoritma dengan menerapkan metode *Corridor First Dungeon Generation* berbasis algoritma *Random Walk* di Unity. Setelah algoritma berhasil diimplementasikan, sistem tersebut diintegrasikan ke dalam gameplay, termasuk penempatan objek interaktif, musuh, dan pengaturan perilaku ruangan yang bersifat dinamis. Terakhir, dilakukan evaluasi terhadap kualitas sistem melalui pengujian menggunakan kuesioner untuk mengukur tingkat *replayability* dan kepuasan pemain terhadap peta yang dihasilkan.

3.2. Tahapan Penelitian



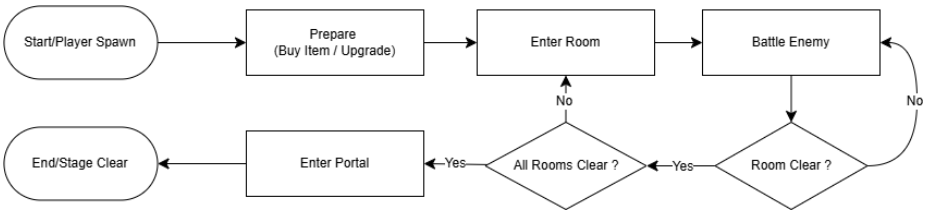
Gambar 1: Grafik tahapan penelitian

Grafik tahapan penelitian ini menggambarkan alur proses yang dilakukan dalam pengembangan dan evaluasi sistem *procedural map* berbasis algoritma *Random Walk* pada game *Scrollbearer*. Tahapan disusun secara berurutan mulai dari analisis kebutuhan hingga ke tahap evaluasi akhir. Setiap langkah saling terhubung secara logis, di mana hasil dari satu tahap menjadi dasar bagi tahap berikutnya

3.2.1. Tahap Pengembangan

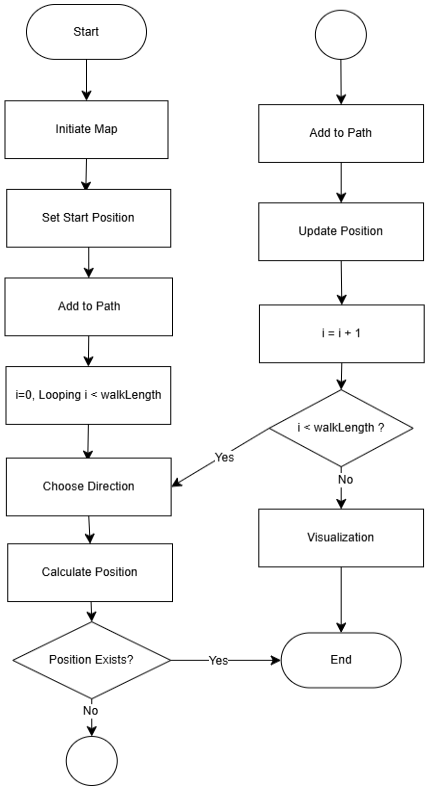
Proses pengembangan dimulai dengan analisis kebutuhan. Dalam tahap ini ditentukan fitur-fitur yang harus tersedia pada game, seperti mekanika dasar gameplay berupa pergerakan karakter, sistem pertarungan, dan interaksi dengan objek serta musuh berbasis NPC. Selain itu, ditentukan pula kebutuhan visual berupa aset 2D, lingkungan 3D, serta elemen audio untuk menciptakan atmosfer permainan. Sistem map procedural menjadi bagian inti yang harus mampu menghasilkan peta dengan struktur koridor, ruangan acak, serta jalur yang tetap terhubung. Penelitian ini juga dilandasi oleh studi literatur terkait algoritma *Random Walk* dan strategi *Corridor First Generation*.

Setelah kebutuhan dirumuskan, dilakukan tahap desain dan perancangan. Alur permainan dirancang agar pemain memulai dari ruangan awal, di mana ia dapat melakukan persiapan seperti membeli item. Setelah itu, pemain akan melanjutkan ke ruangan-ruangan berikutnya yang berisi musuh. Saat memasuki ruangan pertempuran, jalur penghubung ditutup (turun), dan akan terbuka kembali (naik) setelah seluruh musuh dikalahkan. Proses ini terus berlanjut hingga pemain menyelesaikan seluruh ruangan dan menemukan portal akhir permainan.



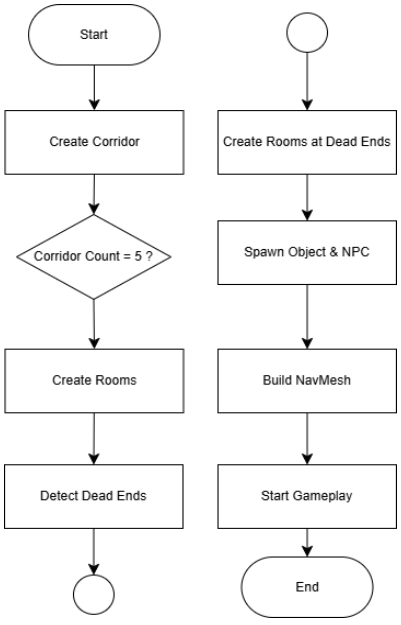
Gambar 2: *Flowchart* Rancangan Alur Permainan

Struktur algoritma yang digunakan terdiri dari dua bagian utama. Pertama adalah algoritma *Random Walk*, yang menghasilkan pola jalur acak melalui proses iteratif dari posisi awal. Setiap langkah berpindah ke arah acak dan ditambahkan ke dalam daftar posisi peta. Kedua adalah algoritma *Corridor First*, yaitu metode yang mengutamakan pembentukan koridor utama terlebih dahulu, kemudian membentuk ruangan di ujung-ujung koridor menggunakan algoritma yang sama. Untuk menghindari jalur yang buntu atau terisolasi, sistem juga secara otomatis menambahkan ruangan tambahan di titik dead end.



Gambar 3: *Flowchart* Algoritma *Random Walk*

Implementasi dilakukan dengan menggunakan beberapa skrip utama di Unity Skrip *CorridorFirstDungeonGenerator3D*.cs bertugas membuat struktur peta berupa koridor dan ruangan. Skrip *CubeVisualizer*.cs digunakan untuk menggambarkan tile peta berdasarkan ketinggian yang ditentukan menggunakan algoritma *perlin noise*. *RoomController*.cs mengatur mekanika musuh dan ruangan, seperti pemicu *event* saat ruangan dibersihkan. Sedangkan *EnvironmentManager*.cs bertugas menambahkan elemen lingkungan seperti pohon dan rumput.



Gambar 4: *Flowchart* Algoritma *Corridor First*

Integrasi antar sistem dilakukan menggunakan fitur *Unity* seperti *NavMesh* untuk mengatur pergerakan NPC dan *event-driven programming* untuk memicu kemunculan portal di akhir ruangan setelah semua musuh dikalahkan. Validasi sistem dilakukan dengan menjalankan pengecekan terhadap jalur yang terputus menggunakan fungsi *FindAllDeadEnds*(), serta memastikan ruangan di titik tersebut tetap terhubung menggunakan *CreateRoomsAtDeadEnd*(). Selain itu, dilakukan simulasi sebanyak 100 kali untuk menguji kestabilan peta dan memastikan tidak terjadi *error* atau kondisi yang membuat peta tidak dapat diselesaikan.

3.2.2. Tahap Evaluasi

Evaluasi dilakukan melalui eksperimen dengan menjalankan game dalam dua mode peta, yaitu peta statis dan peta prosedural. Data dikumpulkan menggunakan kuesioner berbasis skala *Likert* 1–5 untuk mengukur dua aspek utama: *replayability* dan tingkat kepuasan pemain terhadap variasi peta. Selain itu, data juga diperoleh dari *log* sistem yang mencatat hasil variasi struktur peta, jumlah ruangan, panjang koridor, dan posisi objek yang dihasilkan secara otomatis.

Hasil kuesioner kemudian dianalisis secara deskriptif untuk mendapatkan nilai rata-rata dan distribusi skor, serta diuji secara statistik menggunakan uji t-test guna mengetahui signifikansi perbedaan persepsi pemain antara dua mode peta yang digunakan.

3.3. Implementasi Algoritma

Implementasi algoritma dilakukan dengan pendekatan berbasis *grid* 2D yang direpresentasikan menggunakan *Vector2Int*. Posisi awal pemain ditentukan menggunakan variabel *startPosition*.

Pembuatan jalur utama dilakukan dengan menggunakan fungsi *RandomWalkCorridor*(), di mana setiap koridor memiliki panjang 14 langkah. Ruangan kemudian dibuat di setiap ujung koridor menggunakan algoritma *Random Walk*, dengan probabilitas pembentukan ruangan ditetapkan sebesar 100% (*roomPercent* = 1). Agar ukuran ruangan tidak meluas tanpa kendali, digunakan parameter *maxRoomWalkDistance* sebesar 5 unit.

Setiap objek di dalam game diletakkan di posisi yang telah divalidasi menggunakan raycast untuk menghindari tumpang tindih atau posisi yang tidak dapat diakses. Objek utama yang di-*spawn* meliputi NPC di ruangan pertama, patung atau peti di ruangan kedua, air mancur di ruangan ketiga, dan portal di ruangan terakhir.

Mekanika ruangan juga dikontrol melalui sistem animasi, di mana lantai koridor yang memiliki tag *corridor* akan turun saat pemain memasuki ruangan dan naik kembali setelah ruangan selesai dibersihkan. Fungsi yang digunakan dalam mekanika ini adalah *MoveCorridorGroundDown*() dan *MoveCorridorGroundUp*().

4. HASIL DAN PEMBAHASAN

4.1. Hasil Implementasi

4.1.1 Desain dan Implementasi Algoritma *Random Walk*

Implementasi algoritma *Random Walk* dalam game *Scrollbearer* berhasil menghasilkan peta prosedural yang dinamis dan bervariasi. Algoritma ini bekerja dengan memulai dari titik awal tertentu, kemudian bergerak secara acak ke arah kardinal (atas, bawah, kiri, kanan) untuk membentuk jalur koridor dan ruangan.

Peta diinisialisasi sebagai grid dua dimensi dengan titik awal ditentukan oleh variabel *startPosition*. Jalur koridor dibentuk menggunakan metode *RandomWalkCorridor*() yang berjalan sepanjang 14 langkah per koridor. Di ujung koridor, ruangan dibentuk menggunakan algoritma *Random Walk* yang sama, namun dibatasi oleh parameter *maxRoomWalkDistance* agar ukuran ruangan tetap terkendali.

Untuk menangani dead ends, sistem akan otomatis menambahkan ruangan pada posisi jalur buntu agar semua jalur tetap saling terhubung. Beberapa visualisasi hasil implementasi dapat dilihat pada screenshot berikut:



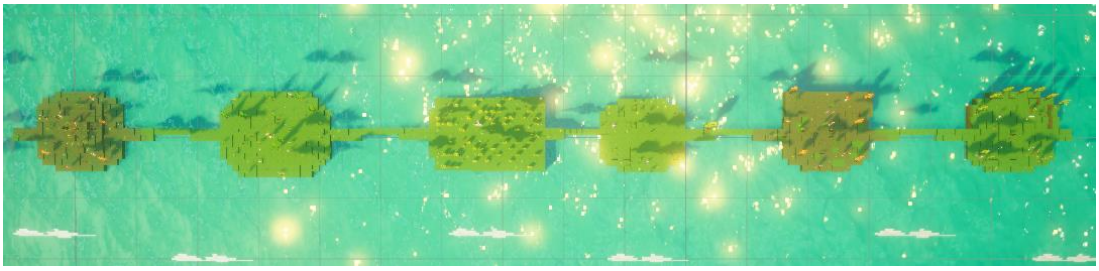
Gambar 5: *Screenshot* Start Menu

Menu utama game menampilkan dua tombol utama: *Play Static* dan *Play Procedura*l, memberikan pemain pilihan untuk memulai permainan dengan mode statis atau prosedural.



Gambar 6: Screenshot In-Game di Mode Statis

Ini adalah tampilan pertama ketika *player* memainkan mode statis, dimana *player* akan langsung *spawn* di area pertama untuk mempersiapkan karakternya memulai permainan.



Gambar 7: *Screenshot Map* di Mode Statis

Contoh tampilan peta di mode statis, di mana layout area tetap sama setiap kali dimainkan, terdiri dari area pertama yang merupakan *preparation* area dan area berikutnya sampai area ke-6 yang berisi musuh untuk diselesaikan.



Gambar 8: *Screenshot In-Game* di *Map* Prosedural

Ini adalah tampilan pertama ketika *player* memainkan mode prosedural, dimana *player* akan langsung *spawn* di area pertama untuk mempersiapkan karakternya memulai permainan.



Gambar 9: Perbedaan *Generate Map* Prosedural 1

Gambar pertama untuk menunjukkan variasi layout peta hasil *generate*, membuktikan bahwa algoritma *Random Walk* mampu menghasilkan peta yang tidak repetitif.



Gambar 10: Perbedaan *Generate Map* Prosedural 2

Gambar kedua dari *generate map* prosedural, menunjukkan variasi yang dihasilkan oleh algoritma *Random Walk*, membuktikan bahwa *map* pada mode prosedural tidak pernah sama .

4.1.2. Penjelasan *Gameplay*

Setelah memilih mode permainan, pemain akan muncul di ruangan pertama. Di sini terdapat NPC merchant yang dapat membantu pemain mempersiapkan diri dengan membeli dan meningkatkan perlengkapan. Selanjutnya, pemain memasuki area pertempuran di mana semua musuh harus dikalahkan sebelum melanjutkan ke ruangan berikutnya. Setelah seluruh ruangan diselesaikan, portal akan muncul untuk mengakhiri permainan.

Mekanika jalan penghubung menjadi bagian penting dari *gameplay*. Ketika pemain masuk ke ruangan yang belum selesai, jalan antar ruangan akan turun menggunakan fungsi *MoveCorridorGroundDown*. Setelah ruangan dibersihkan dari musuh, jalan akan naik kembali melalui *MoveCorridorGroundUp*. Di akhir permainan, sistem memunculkan portal sebagai tanda selesainya seluruh tantangan.

4.1.3. Hasil Kuesioner

Tabel 1 Hasil Kuesioner Pengalaman Bermain

Pertanyaan	Skor rata-rata (1-5)	Contoh data
Kemenarikan Desain Map Statis	3.97	[5, 4, 3, 5, 4]
Kemenarikan Desain Map Prosedural	4.23	[5, 5, 4, 5, 5]
Tingkat Tantangan Mode Statis	3.55	[4, 3, 4, 3, 5]
Tingkat Tantangan Mode Prosedural	4.07	[5, 4, 4, 5, 4]
Kepuasan Eksplorasi di Map Statis	3.78	[4, 3, 4, 4, 3]
Kepuasan Eksplorasi di Map Prosedural	4.34	[5, 5, 4, 5, 5]

Tabel di atas menunjukkan bahwa desain dan eksplorasi pada map prosedural mendapatkan skor lebih tinggi dibandingkan map statis. Hal ini menunjukkan preferensi pemain terhadap variasi dan tantangan yang diberikan oleh peta prosedural.

Tabel 2 Hasil Kuesioner *Replayability*

Pertanyaan	Skor rata-rata (1-5)	Contoh data
Keinginan Memainkan Ulang Mode Statis	3.44	[3,2,4,3,5]
Keinginan Memainkan Ulang Mode Prosedural	4.05	[5,4,4,5,4]
Persepsi Dinamisme Map Prosedural	4.28	[5,5,4,5,5]
Preferensi Pemain (Statis vs Prosedural)	78.9% Prosedural	[Prosedural]

Data yang ditampilkan mencerminkan tanggapan pemain terhadap dua mode permainan. Rata-rata skor dihitung dari keseluruhan jawaban yang diberikan melalui kuesioner, memberikan gambaran umum mengenai persepsi pemain. Selain itu, contoh data individu turut disertakan sebagai ilustrasi untuk memperlihatkan variasi skor yang ada. Terakhir, preferensi pemain ditunjukkan dalam bentuk persentase, yang dihitung berdasarkan jumlah pemain yang memilih mode prosedural dibandingkan dengan mode

statis.

Dari data tersebut dapat disimpulkan bahwa sebagian besar pemain lebih menyukai mode prosedural karena memberikan pengalaman yang tidak repetitif dan membuat mereka ingin terus memainkan ulang game.

4.2. Evaluasi Kualitas Pengalaman Bermain

4.2.1. Tanggapan Pemain terhadap Variasi Peta

Map statis memperoleh skor rata-rata 3.97 untuk desainnya, menunjukkan bahwa struktur tetap dan estetika lingkungan dianggap cukup baik. Namun, mode prosedural mendapat skor lebih tinggi sebesar 4.23 karena pemain menyukai peta yang selalu berubah. Dari sisi tantangan, mode prosedural juga dinilai lebih menantang dengan skor 4.07 dibandingkan 3.55 untuk mode statis.Dalam hal *replayability*, sebanyak 79.5% pemain lebih menyukai mode prosedural. Mereka menyatakan bahwa gameplay menjadi lebih menarik karena variasi peta yang tidak dapat diprediksi.

4.2.2. Saran Pemain untuk Peningkatan

Berdasarkan tanggapan yang dihimpun melalui kuesioner dan observasi selama sesi uji coba, terdapat beberapa saran dari pemain yang dapat dijadikan bahan pertimbangan dalam pengembangan lanjutan mode prosedural. Pertama, pemain mengharapkan adanya peningkatan pada animasi antisipasi serangan musuh. Beberapa responden menyatakan bahwa saat ini serangan musuh terasa tiba-tiba dan sulit diprediksi, sehingga animasi yang lebih jelas sebelum musuh menyerang akan membantu pemain dalam merespons dan menyusun strategi pertahanan.

Selain itu, terdapat masukan mengenai elemen visual dalam medan permainan, khususnya terkait dengan objek seperti pohon dan semak-semak yang terkadang menghalangi pandangan pemain terhadap posisi musuh. Untuk mengatasi hal tersebut, pemain menyarankan agar objek-objek tersebut dibuat lebih transparan atau diberikan indikator visual yang membedakan foreground dan background secara lebih tegas. Hal ini akan meningkatkan keterbacaan lingkungan selama pertarungan berlangsung.

Saran lain yang cukup sering muncul berkaitan dengan aspek estetika. Pemain mengusulkan adanya variasi warna pada peta prosedural, seperti penggunaan warna oranye untuk menciptakan suasana sore atau ungu untuk malam hari. Variasi ini dianggap dapat memperkaya pengalaman visual dan mencegah kejenuhan selama permainan berlangsung.

4.3 Pembahasan

4.3.1 Efektivitas Algoritma *Random Walk*

Algoritma *Random Walk* berhasil menciptakan peta prosedural yang dinamis dan bervariasi, sesuai dengan tujuan penelitian. Peta yang dihasilkan tidak hanya memberikan variasi visual, tetapi juga meningkatkan *replayability game*. Hal ini terlihat dari tanggapan positif pemain terhadap mode prosedural, yang dinilai lebih menarik dan menantang dibandingkan mode statis.

4.3.2 Pengaruh Variasi Peta terhadap Pengalaman Bermain

Mode prosedural memberikan pengalaman bermain yang lebih dinamis dan tidak repetitif. Pemain merasa lebih tertantang karena mereka tidak dapat menghafal layout peta seperti pada mode statis. Selain itu, mekanisme jalan penghubung yang turun dan naik menambah elemen strategis dalam *gameplay*.

4.3.3 Implikasi untuk *Game Roguelike*

Implementasi algoritma *Random Walk* dalam game *Scrollbearer* membuktikan bahwa *procedural generation* dapat menjadi solusi efektif untuk meningkatkan *replayability* dan pengalaman

bermain dalam genre roguelike. Dengan penyesuaian lebih lanjut, seperti penambahan fitur *random event* dan variasi *biome*, potensi algoritma ini dapat dioptimalkan lebih jauh.

5. SIMPULAN DAN SARAN

Dari hasil implementasi dan evaluasi yang telah dilakukan, dapat disimpulkan bahwa algoritma *Random Walk* mampu diterapkan secara efektif dalam game *Scrollbearer* untuk menghasilkan peta prosedural yang dinamis dan bervariasi. Pola jalur yang terbentuk bersifat acak namun tetap terhubung, berkat integrasi dengan metode *Corridor First* dan mekanisme penanganan *dead ends*. Kombinasi tersebut memastikan bahwa struktur peta tidak hanya variatif, tetapi juga bebas dari area terisolasi. Lebih lanjut, algoritma ini berkontribusi nyata dalam meningkatkan aspek *replayability* dan pengalaman bermain secara keseluruhan. Hasil kuesioner menunjukkan bahwa 79,5% responden lebih memilih mode prosedural dibandingkan mode statis. Hal ini diperkuat oleh skor kepuasan eksplorasi sebesar 4,34 dan tingkat tantangan sebesar 4,07, yang menunjukkan bahwa pemain merasa lebih terlibat dalam permainan ketika peta yang dihadirkan bersifat tidak terduga. Dengan demikian, pendekatan *procedural generation* berbasis *Random Walk* terbukti relevan dan efektif dalam konteks game bergenre *roguelike* seperti *Scrollbearer*. Keunikan layout yang dihasilkan pada setiap permainan memberikan nilai tambah yang signifikan dalam menjaga ketertarikan pemain.

Sebagai tindak lanjut dari penelitian ini, terdapat beberapa arah pengembangan yang dapat dilakukan untuk menyempurnakan pengalaman bermain. Dari sisi visual dan mekanik, peningkatan pada animasi antisipasi serangan musuh perlu dipertimbangkan agar pemain dapat membaca pola serangan lebih mudah. Selain itu, implementasi indikator visual atau transparansi pada objek penghalang pandangan akan membantu menciptakan medan pertempuran yang lebih terbuka dan strategis. Variasi warna pada peta juga direkomendasikan untuk meningkatkan imersi dan estetika. Pada sisi teknis algoritma, penggabungan algoritma *Random Walk* dengan pendekatan *procedural* lainnya berpotensi menciptakan *biome* atau tema peta yang lebih kompleks. Penambahan elemen kejutan seperti jebakan tersembunyi, harta karun, atau NPC unik dapat memperkaya elemen eksplorasi dan strategi. Evaluasi jangka panjang juga penting untuk dilakukan, termasuk pengujian dengan jumlah partisipan yang lebih banyak untuk menilai potensi *burnout* pemain terhadap konten prosedural yang berulang. Sistem kesulitan adaptif yang menyesuaikan tingkat tantangan berdasarkan performa pemain juga patut dipertimbangkan, misalnya melalui penyesuaian jumlah musuh atau distribusi item. Akhirnya, pemanfaatan teknologi seperti kecerdasan buatan atau machine learning dalam *procedural generation* dapat membuka peluang baru. Teknologi ini memungkinkan penciptaan ruang permainan dengan tema khusus berdasarkan preferensi pemain atau riwayat permainan mereka, sehingga pengalaman bermain menjadi lebih personal dan mendalam. Dengan menerapkan berbagai saran ini, diharapkan *Scrollbearer* dapat terus dikembangkan menjadi game *roguelike* yang tidak hanya menantang dan variatif, tetapi juga memberikan pengalaman bermain yang semakin menarik dan kompetitif.

DAFTAR PUSTAKA

[1] J. Togelius, N. Shaker, and M. J. Nelson, “Introduction,” in *Procedural Content Generation in Games*, N. Shaker, J. Togelius, and M. J. Nelson, Eds., Cham: Springer International Publishing, 2016, pp. 1–15. doi: 10.1007/978-3-319-42716-4_1.

[2] M. Hendrikx, S. Meijer, J. Velden, and A. Iosup, “Procedural Content Generation for Games: A Survey,” *ACM Trans. Multimed. Comput. Commun. Appl. TOMCCAP*, vol. 9, Feb. 2013, doi: 10.1145/2422956.2422957.

[3] T. Short and T. Adams, *Procedural Generation in Game Design*. CRC Press, 2017.

[4] B. Bucklew, “Algorithms and Approaches,” in *Procedural Generation in Game Design*, A K Peters/CRC Press, 2017.

[5] A. Koesnaedi and W. Istiono, “Implementation Drunkard’s Walk Algorithm to Generate Random Level in Roguelike Games,” *Int. J. Multidiscip. Res. Publ.*, vol. 5, no. 2.

[6] J. Baron, “Procedural Dungeon Generation Analysis and Adaptation,” presented at the Conference: the SouthEast Conference, Apr. 2017, p. 171. doi: 10.1145/3077286.3077566.

[7] T. Lacoma, “What is Unity?,” Android Police. Accessed: Mar. 05, 2025. [Online]. Available: https://www.androidpolice.com/what-is-unity/

[8] X. Chen, M. Xu, and Y. An, “Identifying the essential nodes in network pharmacology based on multilayer network combined with random walk algorithm,” *J. Biomed. Inform.*, vol. 114, p. 103666, Feb. 2021, doi: 10.1016/j.jbi.2020.103666.

[9] K. Berahmand, E. Nasiri, S. Forouzandeh, and Y. Li, “A preference random walk algorithm for link prediction through mutual influence nodes in complex networks,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 8, Part A, pp. 5375–5387, Sep. 2022, doi: 10.1016/j.jksuci.2021.05.006.

[10] M. Ijai, A. Arbansyah, and S. H. Suryawan, “Penerapan Algoritma Random Walk Untuk Procedural Map Pada Game ‘The Last Hope’ 2D,” *J. CoSciTech Comput. Sci. Inf. Technol.*, vol. 4, no. 3, Art. no. 3, 2023, doi: 10.37859/coscitech.v4i3.6420.

[11] G. M. Muktadir, A. Jawad, I. Paranjape, J. Whitehead, and A. Shepelev, “Procedural Generation of High-Definition Road Networks for Autonomous Vehicle Testing and Traffic Simulations,” *SAE Int. J. Connect. Autom. Veh.*, vol. 6, no. 1, pp. 99–120, May 2022, doi: 10.4271/12-06-01-0007.

[12] T. Sentausa, “Game Roguelike Itu Apa? Ayo Ungkap Apa Arti Genre Roguelike!,” UrbanIdea ID. Accessed: Mar. 04, 2025. [Online]. Available: https://urbanidea.id/game/game-roguelike-adalah/

[13] C. Fan, D. Li, Y. Teng, D. Fan, and G. Ding, “Exploiting Non-visible Relationship in Link Prediction Based on Asymmetric Local Random Walk,” in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds., Cham: Springer International Publishing, 2017, pp. 731–740.

[14] R. Adellin, C. T. Khuan, and L. D. Gertrude, “Conceptual Framework Puzzle Game with High Replayability,” *J. Phys. Conf. Ser.*, vol. 1228, no. 1, p. 012070, May 2019, doi: 10.1088/1742-6596/1228/1/012070.

[15] K. Perlin, “An image synthesizer,” *SIGGRAPH Comput Graph*, vol. 19, no. 3, pp. 287–296, Jul. 1985, doi: 10.1145/325165.325247.

[16] R. Likert, “A technique for the measurement of attitudes,” *Arch. Psychol.*, vol. 22 140, pp. 55–55, 1932.

[17] H. Boone and D. Boone, “Analyzing Likert Data,” *J. Ext.*, vol. 50, no. 2, Apr. 2012, doi: 10.34068/joe.50.02.48.

[18] Z. B. R. Ghani, “Apa itu NPC dan MC dalam Game? Istilah Viral di Media Sosial,” Duniaku. Accessed: Mar. 05, 2025. [Online]. Available: https://duniaku.idntimes.com/game/konsol-pc/zihan-berliana-ram-ghani/apa-itu-npc

[19] R. Sheldon, “What is the C# object-oriented programming language? – TechTarget Definition,” WhatIs. Accessed: Mar. 05, 2025. [Online]. Available: https://www.techtarget.com/whatis/definition/C-Sharp