PENGEMBANGAN LINGKUNGAN DINAMIS TIGA DIMENSI PADA GAME ENDLESS RUNNER BERBASIS UNITY DENGAN IMPLEMENTASI PERLIN NOISE

Alvin Fredericco[*], Oddy Virgantara Putra[2]
[1,2] Jurusan Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Darussalam Gontor
Jl. Raya Siman, Dusun I, Demangan, Kec. Siman, Kabupaten Ponorogo, Jawa Timur 63472
Email : alvinfredericco85@student.cs.unida.gontor.ac.id
Email: oddy@unida.gontor.ac.id

Abstrak

Permainan dengan kategori Endless Runner sangat bergantung pada Procedural Content Generation (PCG) untuk menjamin variasi dan kemampuan bermain ulang yang tak terbatas. Studi ini bertujuan untuk merancang sistem lingkungan tiga dimensi (3D) yang adaptif dan efisien pada mesin game Unity. Sistem yang dikembangkan mengintegrasikan Perlin Noise untuk penempatan objek lingkungan dan sistem tile-looping untuk mendaur ulang area permainan secara efisien. Batasan studi ini adalah pada generasi variasi permukaan lingkungan horizontal (sumbu X-Z), dengan fokus pada penggunaan Perlin Noise 3D untuk memvariasikan elevasi (Y-axis). Karakteristik spesifik dari algoritma yang digunakan adalah penyesuaian skala (scale) dan amplitudo (amplitude) yang diatur secara iteratif untuk menghasilkan kontur terrain yang beragam. Hasil uji coba menunjukkan sistem mampu menyajikan lingkungan dinamis dengan variasi tinggi, meminimalisir pola repetisi , dan penggunaan tile-looping terbukti menjaga penggunaan memori tetap stabil dan mencegah penurunan performa.

***Kata kunci**: Endless Runner, Perlin noise, Procedural content generation, Tile-looping, Unity*

1. PENDAHULUAN

Pengembangan *game* merupakan bidang yang mengalami perkembangan pesat, di mana penciptaan lingkungan (*environment*) menjadi elemen krusial untuk menghadirkan pengalaman interaktif. Pada *genre endless runner*, interaksi pemain sangat bergantung pada variasi lingkungan yang muncul secara terus-menerus. Namun, pengembangan peta secara manual sering kali membutuhkan sumber daya yang besar dan berisiko menghasilkan pola yang repetitif **(**Andrian dkk, 2023).

Procedural Content Generation (PCG) menjadi pendekatan umum dalam menghasilkan konten permainan secara otomatis melalui pemrosesan algoritmik (Atthariq dkk., 2022). Namun, penerapan PCG dalam lingkungan 3D memiliki tantangan teknis, terutama terkait kualitas variasi dan biaya komputasi. Penggunaan *random noise* konvensional sering menghasilkan perubahan yang kasar, sementara proses generasi konten secara terus-menerus dapat menyebabkan penumpukan objek, peningkatan konsumsi memori, dan penurunan performa. Perlin Noise menawarkan keunggulan berupa *gradient noise* yang menghasilkan variasi lebih halus dan terkontrol, sehingga sesuai digunakan untuk pengaturan bentuk dan distribusi lingkungan (Ahmed & Pandey, 2023).

Implementasi PCG pada permainan tanpa batas juga memerlukan strategi manajemen memori yang baik. Tanpa mekanisme pengaturan ulang objek, proses generasi segmen yang berlangsung terus-menerus dapat menyebabkan pemborosan memori, *stuttering*, atau penurunan frame rate. Beberapa penelitian sebelumnya telah memanfaatkan Perlin Noise untuk generasi terrain, tetapi belum banyak yang menggabungkannya dengan teknik manajemen memori yang efisien untuk skenario gameplay tanpa batas (Juwianto dkk., 2023; Selviana dkk., 2023). Hal ini menunjukkan adanya kebutuhan untuk pengembangan sistem yang tidak hanya mampu menghasilkan variasi lingkungan, tetapi juga memastikan stabilitas performa dalam jangka panjang.

Penelitian ini mengusulkan integrasi algoritma Perlin Noise dengan sistem *tile-looping* sebagai solusi. Perlin Noise digunakan untuk menghasilkan variasi lingkungan secara halus, sedangkan tile-looping berfungsi mendaur ulang segmen peta agar jumlah objek yang aktif tetap terkendali.

2. METODOLOGI

Penelitian ini menggunakan metode *Game Development Life Cycle* (GDLC) yang terdiri dari fase inisiasi, pra-produksi, produksi, pengujian, beta, dan rilis (Agung Saputra dkk. 2022). Fokus utama pembahasan pada makalah ini terletak pada tahap produksi, khususnya implementasi algoritma.

2.1. Implementasi Perlin Noise

Algoritma *Perlin Noise* diimplementasikan sebagai inti dari mekanisme Procedural Content Generation (PCG) untuk menghasilkan kontur *terrain* dan menentukan penempatan objek lingkungan secara prosedural. Pemilihan algoritma ini didasarkan pada kemampuannya dalam menciptakan pola acak yang kohesif dan tampak alami (*smooth gradient noise*), yang krusial untuk meminimalisir pola repetitif.

Penggunaan *Perlin Noise* mengacu pada pendekatan yang telah digunakan dalam penelitian sebelumnya, yang menunjukkan efektivitas *gradient noise* dalam membentuk pola alami (Latif dkk, 2022; Sabihartono, 2023). Referensi ini digunakan sebagai justifikasi pemilihan metode, bukan sebagai uraian teoritis, guna memastikan bahwa teknik yang diterapkan memiliki dasar empiris yang relevan.

Pada proses implementasikan dalam fungsi GetRandomTileIndexWithLightPerlin(). Nilai dasar diambil dari fungsi Mathf.PerlinNoise yang dipengaruhi oleh tileCounter (urutan tile) dan noiseScale (skala frekuensi). Nilai ini kemudian dimodifikasi dengan variabel bias acak untuk memberikan variasi lokal. Rumus penentuan indeks *tile* (I_{tile}) adalah sebagai berikut (1):

$$N_{base} = \text{Mathf.PerlinNoise}(C \cdot S, \text{Seed})$$
$$V_{mix} = \text{Clamp01}\big(N_{base} + \text{Random.Range}(-0.3, 0.3)\big)$$
$$I_{tile} = \text{RoundToInt}\big(\text{Lerp}(I_{min}, I_{max}, V_{mix})\big)$$

Dimana:

1) C ($tileCounter$): Variabel inkremental yang merepresentasikan posisi langkah pemain.
2) S ($noiseScale$) : Konstanta (0.1f) untuk mengatur kehalusan transisi antar jenis tile.
3) $Seed$: Nilai acak yang diinisialisasi saat Start untuk memastikan setiap sesi permainan memiliki pola unik.
4) I_{min}, I_{max} : Rentang indeks prefab yang tersedia dalam antrean aset.

2.2. Sistem Tile-Looping

Sistem manajemen memori diimplementasikan menggunakan struktur data antrean (*queue*) yang memantau posisi pemain secara *real-time*. Mengingat sifat permainan *endless runner*, efisiensi memori dijaga dengan membatasi jumlah objek yang dirender di layar menggunakan variabel bufferTiles.

Pemilihan tile dilakukan dengan menggabungkan nilai Perlin Noise dan sedikit bias acak (randomBias) untuk menentukan indeks prefab yang akan dimunculkan. Pendekatan ini memastikan variasi visual tetap terkontrol dan tidak repetitif. Penggunaan Perlin Noise sebagai dasar variasi mengikuti praktik yang telah terbukti efektif dalam menciptakan pola yang natural untuk sistem generatif (Ramadhan & Indriyanti, 2022).

Sistem membatasi jumlah objek yang dirender di layar menggunakan variabel bufferTiles. Mekanisme ini bekerja dalam siklus *First-In-First-Out* (FIFO) sebagai berikut:

1. Ketika pemain bergerak maju dan jarak (z_{player}) mendekati batas area generasi (z_{spawn}), fungsi $SpawnTile()$ dipanggil. Sistem melakukan instansiasi (Instantiate) prefab baru berdasarkan indeks yang dipilih oleh algoritma Perlin Noise.
2. Penyimpanan Referensi: Setiap objek *tile* yang baru dibuat dimasukkan ke dalam antrean aktif ($active.Enqueue(tile)$).
3. Pembersihan Memori (*Garbage Collection Assistant*) : Pada setiap *frame*, sistem memeriksa jumlah elemen dalam antrean. Jika jumlah *tile* aktif melebihi batas *buffer* ($active.Count >$

$bufferTiles$), *tile* paling lama (paling belakang) dikeluarkan dari antrean ($Dequeue$) dan segera dihapus dari memori menggunakan fungsi $Destroy()$.

Pendekatan ini memastikan bahwa meskipun permainan berjalan tanpa batas waktu, beban memori (RAM) tetap konstan karena jumlah objek yang aktif pada satu waktu selalu dibatasi secara deterministik, mencegah terjadinya kebocoran memori (*memory leak*).

3. HASIL DAN PEMBAHASAN

Implementasi algoritma Perlin Noise berhasil menciptakan variasi lingkungan yang dinamis pada setiap sesi permainan. Berdasarkan nilai *noise* yang dihasilkan, sistem menempatkan **objek rintangan** dan dekorasi pada koordinat yang berbeda-beda secara otomatis. Hal ini mencegah pemain menghafal pola level (*pattern memorization*), sehingga tantangan permainan tetap terjaga.

Gambar 1 Visualisasi hasil generasi

Visualisasi hasil generasi dapat dilihat pada **Gambar 1**, di mana rintangan tersebar di sepanjang jalur permainan di antara elemen dekoratif lainnya. Penempatan ini divalidasi melalui pengujian fungsional untuk memastikan rintangan tidak menumpuk secara mustahil (*impossible path*) dan deteksi tabrakan (*collision detection*) berfungsi akurat saat karakter mengenai objek tersebut.

3.1. Analisis Performa Tile-Looping

Penerapan sistem *tile-looping* terbukti krusial dalam menjaga stabilitas aplikasi. Tanpa sistem ini, instansiasi objek yang terus-menerus akan membebani memori (*memory leak*) dan menyebabkan penurunan *frame rate*. Dengan menerapkan logika daur ulang objek (*object pooling*), objek yang sudah tidak terlihat oleh kamera tidak dihancurkan, melainkan dinonaktifkan dan digunakan kembali di depan jalur pemain. Hasil pengujian menunjukkan bahwa mekanisme ini berhasil menjaga beban komputasi tetap ringan, sehingga *game* dapat berjalan lancar (*smooth*) secara terus-menerus tanpa batasan waktu.

Tabel 1 Hasil pengujian Performa Hardware

No	Pengujian	Hasil
1	Frame Rate (FPS)	300-550
2	GPU utilization	± 90%
3	CPU utilization	± 21%
4	Memory Used (VRAM)	± 4%

3.2. Implementasi Kontrol Karakter

Penelitian ini menerapkan kontrol berbasis *keyboard* sebagai input utama untuk memastikan gerakan dalam menghindari rintangan yang digenerasi secara prosedural. Pemetaan kontrol dirancang mengikuti standar navigasi *game* pada umumnya (*WASD* atau *Arrow Keys*) untuk meminimalkan kurva pembelajaran (*learning curve*) pemain. Tombol 'W' atau 'Panah Atas' berfungsi untuk melompat (*jump*), 'S' atau 'Panah Bawah' untuk menunduk (*slide*), serta 'A' dan 'D' atau panah samping untuk berpindah jalur.

3.3. Pengujian Fungsional (Blackbox)

Pengujian *blackbox* dilakukan untuk memvalidasi fungsi utama sistem generasi peta. Pada **Tabel 1** hasil pengujian menunjukkan bahwa seluruh fungsi prosedural berjalan sesuai rancangan, di mana algoritma mampu menghasilkan variasi *terrain* yang natural dan sistem *looping* bekerja tanpa *error* kritikal.

Tabel 2 Pengujian blackbox

No	Pengujian	Hasil
1	Sistem procedural generation (Perlin Noise) menghasilkan variasi terrain yang natural dan tidak repetitif.	Berhasil
2	Sistem tile-looping menciptakan efek dunia tak terbatas tanpa terasa monoton.	Berhasil
3	Sistem collision detection pada rintangan bekerja dengan akurat	Berhasil

3.4. Diskusi dan keterbatasan penelitian

Hasil penelitian menunjukkan bahwa integrasi algoritma Perlin Noise efektif dalam mengatasi masalah repetisi visual pada genre endless runner, di mana transisi lingkungan yang dihasilkan terbukti lebih natural dan halus dibandingkan metode pengacakan linier. Temuan ini sejalan dengan penelitian terbaru yang menekankan pentingnya variasi biome dalam procedural generation untuk meningkatkan pengalaman eksplorasi (Wahyudi dkk, 2025). Keberhasilan visual ini didukung oleh stabilitas performa sistem tile-looping, yang memungkinkan permainan berjalan tanpa batas waktu tanpa mengalami penurunan frame rate (lag). Selain itu, evaluasi pada gameplay mengonfirmasi bahwa rintangan yang digenerasi secara prosedural tetap proporsional dan dapat dinavigasi dengan baik.

Namun demikian, penelitian ini memiliki keterbatasan pada variasi perangkat keras (hardware) yang diuji. Evaluasi performa saat ini terbatas pada perangkat uji yang tersedia (Intel Core i7-13700HX dan GPU NVIDIA RTX 4060), dan belum mencakup pengujian pada perangkat dengan spesifikasi rendah atau mobile karena batasan lingkup penelitian. Meskipun demikian, indikator penggunaan VRAM yang sangat rendah dan stabil (±4%) selama pengujian menunjukkan potensi kuat bahwa algoritma ini memiliki overhead memori yang ringan, sehingga secara teoritis dapat diimplementasikan pada perangkat dengan spesifikasi lebih rendah tanpa kendala performa yang signifikan.

Sebagai saran untuk pengembangan selanjutnya, penelitian ini dapat diperluas dengan melakukan optimasi dan pengujian pada perangkat mobile untuk memvalidasi efisiensi algoritma pada spesifikasi rendah. Selain itu, metode interaksi permainan dapat dikembangkan agar lebih imersif melalui implementasi kontrol berbasis suara (voice control) atau deteksi gestur tubuh (body control) seperti penelitian sebelumnya (Jaman dan Fergina 2021; Ghani dkk, 2024) guna melengkapi mekanisme navigasi standar (keyboard) yang digunakan saat ini.

4. KESIMPULAN

Berdasarkan hasil penelitian, implementasi algoritma *Perlin Noise* berhasil menghasilkan lingkungan 3D yang variatif dan natural pada *game endless runner*, sehingga menghindari pola repetitif yang membosankan. Sementara itu, integrasi sistem *tile-looping* terbukti efektif dalam

menciptakan ilusi dunia tanpa batas. Mekanisme daur ulang *tile* ini memastikan kinerja aplikasi tetap optimal dan stabil selama *runtime*, menjadikan metode ini referensi yang tepat untuk pengembangan *game* prosedural yang efisien dalam manajemen memori.

Sebagai saran untuk pengembangan selanjutnya, penelitian ini dapat diperluas dengan melakukan optimasi dan pengujian pada perangkat *mobile* untuk memvalidasi efisiensi algoritma pada spesifikasi rendah**.** Selain itu, metode interaksi permainan dapat dikembangkan agar lebih imersif melalui implementasi kontrol berbasis suara (*voice control*) atau deteksi gestur tubuh (*body control*) seperti penelitian sebelumnya (Jaman dan Fergina 2021; Ghani dkk. 2024), guna melengkapi mekanisme navigasi standar (*keyboard*) yang digunakan saat ini.

DAFTAR PUSTAKA

Juwiantho, Hans, Liliana Liliana, dan Michael Budiono. "Procedural Content Generation pada Game Tower Defense menggunakan Perlin Noise dan Algoritma Floyd Warshall." *Journal of Animation and Games Studies* 9, no. 1 (2023): 8–15.

Selviana, Renita, dan Dauw Bastha Fiastat Lugata. "Implementasi Generate Map dan Pemunculan Objek secara Acak pada Game 3D Menggunakan Bahasa C# dan Metode Perlin Noise di Unity: Studi Kasus pada Game Development." *Jurnal Spirit* 15, no. 1 (2023): 18–22.

Shen, Zhenyuan. "Procedural Generation in Games: Focusing on Dungeons." *SHS Web of Conferences* 144 (2022): 02005.

Latif, G., Alghazo, A., Kazimi, R., Samara, N., Alghazo, B., & Alghazo, M. "A Critical Evaluation of Procedural Content Generation Approaches for Digital Twins." *J. Sens.*, 2022, Article ID 5629645. doi:10.1155/2022/5629645.

Ramadhan, D. A., & Indriyanti, A. D. "Procedural Content Generation pada Game World Exploration Sandbox Menggunakan Algoritma Perlin Noise." *J. Inform. Comput. Sci. (JINACS)*, 4(1), 86–91, 2022.

Sabihartono, I. Y. "Analisis Pemanfaatan Perlin Noise dalam Pembuatan Terrain Game Virtual Biotope Menggunakan Metode Procedural Content Generation." Skripsi S1, Universitas Pendidikan Indonesia, Bandung, 2023.

Agung Saputra, A., Nonggala Putra, F. & Darma Rusdian Yusron, R. (2022). *Pembuatan Game Edukasi Pengenalan Kebudayaan Indonesia Menggunakan Metode Game Development Life Cycle (GDLC) Berbasis Android.*

Atthariq, M. H., Eridani, D., & Fauzi, A. *Implementasi Procedural Content Generation pada Game Menggunakan Unity*. Jurnal Teknik Komputer, 1(2), 62–72. Available at: https://ejournal3.undip.ac.id/index.php/jtk.

Ghani, M. A., Pohan, A. B., Gunawan, D., & Saputra, Y. (2024). "Penerapan Game Edukasi 3D Endless Runner Berbasis Android Sebagai Media Belajar Matematika Anak." *Infotek: Jurnal Informatika dan Teknologi* 7(1), 288–298. doi: 10.29408/jit.v7i1.24194.

Jaman, A. B., & Fergina, A. (2021). *Implementasi Speech Recognition Berbasis Android dalam Optimalisasi Komunikasi bagi Penyandang Tunarungu.*

Wahyudi, A., Pawelloi, A. I., & Sirimorok, N. (2025). "Implementasi Procedural Generation Pada Game Aksi Survival." *Jurnal Informatika dan Teknologi Pendidikan* 5(2), 144–151. doi: 10.59395/jitp.v5i2.142.