



Jurnal SANTI (Sistem Informasi dan Teknologi Informasi)
Vol. 6 No. 2 Tahun 2026.
DOI : <https://doi.org/10.58794/santi.v6i2.2434>

Otomatisasi Tata Letak Aset Lingkungan Secara Real-Time Berbasis Wave Function Collapse pada Game Endless Runner 3D

Weli Andrian^{*1}, Yandiko Saputra Sy², Salamun³

^{1,3}Teknik Informatika, Fakultas Teknik, Universitas Abdurrab, ²Ilmu Komputer, Fakultas
Matematika dan Ilmu Pengetahuan Alam, Universitas Riau

e-mail: ^{*}weli@univrab.ac.id, ²yandiko.saputra@lecturer.unri.ac.id, ³salamun@univrab.ac.id

Abstrak

Procedural Content Generation (PCG) pada game endless runner sering dihadapkan pada tantangan ganda, yakni kejenuhan pola statis serta tingginya beban komputasi apabila menggunakan generasi level 3D yang murni acak. Penelitian ini mengusulkan sebuah arsitektur pipeline otomatisasi yang mengadaptasi logika algoritma Wave Function Collapse (WFC) ke dalam lingkungan spasial 3D linier secara real-time. Berbeda dengan pra-komputasi WFC konvensional, sistem ini memanfaatkan pendekatan sliding window grid yang dikombinasikan dengan C# Coroutines asinkron untuk merakit segmen (chunks) lingkungan tanpa memicu penurunan frame rate. Modul Pengendali Daur Ulang Memori (Dynamic Object Pooling Manager) diintegrasikan untuk memitigasi fragmentasi memori dan mengoptimalkan operasi Garbage Collection. Evaluasi sistem mencakup kinerja komputasi, validasi topologi spasial, dan metrik pengalaman pemain. Hasil eksperimen menunjukkan rata-rata waktu generasi sebesar 35,1 ms per segmen, mempertahankan kestabilan 60 FPS pada tingkat kepadatan aturan (constraints) rendah hingga menengah. Implementasi Socket Registry untuk penentuan aturan ketetanggaan menghasilkan akurasi transisi spasial sebesar 99,43% dan tingkat keandalan navigasi lintasan sebesar 99,73%, yang secara efektif meminimalkan jalur buntu struktural (dead-ends). Evaluasi pengguna mengonfirmasi adanya peningkatan signifikan pada estetika visual dan tingkat keinginan bermain ulang.

Kata kunci: Wave Function Collapse, Procedural Content Generation, Endless Runner, Pemrosesan Asinkron, Unity 3D

Abstract

Procedural Content Generation (PCG) in endless runner games often faces the dual challenges of static pattern fatigue and excessive computational overhead when employing purely randomized 3D level generation. This research proposes an automated pipeline architecture that adapts the logic of the Wave Function Collapse (WFC) algorithm into a real-time, linear 3D spatial environment. Unlike conventional pre-computed WFC, this system utilizes a sliding window grid approach combined with asynchronous C# Coroutines to assemble environmental chunks on-the-fly without triggering frame rate degradation. Furthermore, a Dynamic Object Pooling Manager is integrated to mitigate memory fragmentation and optimize Garbage Collection operations. The system's evaluation encompassed computational performance, spatial topology validation, and player experience metrics. Experimental results demonstrate an average generation time of 35.1 ms per chunk, maintaining a stable 60 FPS under low-to-medium constraint densities. The implementation of a Socket Registry for adjacency rules yielded a spatial

transition accuracy of 99.43% and a path navigability rate of 99.73%, effectively minimizing structural dead-ends. User evaluations confirmed significant improvements in visual aesthetics and replayability.

Keywords: *Wave Function Collapse, Procedural Content Generation, Endless Runner, Asynchronous Processing, Unity 3D.*

1. Pendahuluan

Perkembangan industri permainan video saat ini telah mendorong inovasi yang pesat dalam berbagai genre, salah satunya adalah genre *endless runner*. Permainan jenis ini sangat diminati karena mekanismenya yang sederhana namun menantang, di mana karakter dituntut untuk terus berlari melewati rintangan yang muncul tanpa henti. Selain sebagai media hiburan, genre *endless runner* juga telah banyak diadopsi dalam lingkungan akademik dan edukasi, seperti pemanfaatannya sebagai inovasi media pembelajaran guna meningkatkan literasi matematika siswa [2] serta pengembangan *game* edukasi pengenalan tanaman obat [6]. Aksesibilitas genre ini semakin meluas dengan adanya pengembangan pada berbagai *platform*, baik itu *platform mobile* maupun berbasis *web* [5].

Hal utama dari mekanika permainan *endless runner* adalah kemampuannya untuk menghasilkan level yang tidak pernah habis. Untuk mencapai hal ini, pengembang sangat bergantung pada teknik *Procedural Content Generation* (PCG). PCG memungkinkan sistem untuk membangun konten permainan secara otomatis melalui algoritma komputasi, sehingga mengurangi beban kerja desainer dalam merancang panjang level secara manual [13]. Teknik ini telah menjadi fondasi utama yang terbukti efektif sejak era pengembangan *game platformer* 2D [4], namun seiring dengan transisi standar industri menuju lingkungan 3D yang lebih kompleks, kebutuhan akan metode PCG yang lebih cerdas, variatif, dan terstruktur menjadi sangat krusial.

Dalam pengembangan tata letak aset spasial tiga dimensi, penerapan PCG tradisional sering kali menemui kendala berupa penempatan objek yang saling tumpang tindih (*clipping*), berongga, atau tidak memiliki kohesi visual. Generasi konten prosedural dalam ruang 3D menuntut algoritma yang tidak hanya memunculkan objek secara acak, tetapi juga memiliki pemahaman terhadap aturan ruang agar lingkungan yang dihasilkan tetap masuk akal dan dapat dilalui oleh pemain [12]. Tantangan rekayasa ini semakin esensial ketika sistem ditugaskan untuk melakukan otomatisasi tata letak aset lingkungan dalam skala besar (*large-scale content generation*), di mana algoritma harus mampu menjaga stabilitas struktur berantai tanpa membebani kinerja komputasi pada memori sistem [8].

Sebagai solusi atas problematika desain tata letak tersebut, algoritma *Wave Function Collapse* (WFC) muncul sebagai salah satu metode PCG yang memiliki perkembangan yang baik dalam riset multimedia terkini. Terinspirasi dari konsep mekanika kuantum terkait keruntuhan fungsi gelombang dan pemecahan simetri spontan (*spontaneous symmetry breaking*), WFC bekerja dengan cara membatasi probabilitas penempatan aset berdasarkan aturan atau *constraints* keterhubungan objek yang telah didefinisikan sebelumnya [15]. Implementasi algoritma WFC memastikan bahwa setiap modul atau aset lingkungan yang dirakit memiliki tingkat transisi yang valid, menjadikannya sangat ideal untuk merancang arsitektur permainan yang logis secara estetika dan fungsional [9].

Keandalan WFC telah memicu berbagai pengembangan dan modifikasi di ranah akademik untuk menangani pemodelan yang lebih rumit [11]. Salah satu modifikasi yang signifikan adalah penggunaan struktur penyelesaian berbasis graf (*graph-based*) yang memungkinkan WFC untuk merakit konten dengan topologi yang dinamis tanpa harus bergantung murni pada batasan sistem *grid* ortogonal [3]. Lebih lanjut, perumusan cara kerja algoritma ini ke dalam kerangka kerja analitis rantai Markov (*Markovian framing*) telah memberikan landasan probabilistik yang lebih

solid dalam memprediksi pola tata letak agar variasi level dapat dirakit secara harmonis dari awal permainan hingga akhir [1].

Tren riset mutakhir juga mengindikasikan pergeseran ke arah hibridisasi algoritma WFC dengan model kecerdasan buatan (*Machine Learning*). Pemanfaatan *Reinforcement Learning* untuk membimbing algoritma WFC terbukti mampu menciptakan teknik *Procedural Content Generation* yang lebih adaptif dan dapat dikendalikan (*controllable*), di mana agen AI mempelajari komposisi terbaik guna membangun lintasan level yang tingkat kesulitannya terukur bagi pemain [7], [10]. Ke depannya, kompleksitas PCG dalam merakit lingkungan permainan bahkan mulai diarahkan pada arsitektur komputasi kuantum (*quantum algorithms*) untuk menyelesaikan masalah *constraint-solving* dengan latensi seminimal mungkin [14].

Tinjauan terhadap berbagai literatur dan penelitian terdahulu menunjukkan bahwa sebagian besar pendekatan *Procedural Content Generation* (PCG) pada *game endless runner* masih sangat bergantung pada teknik pengacakan buta (*blind randomization*) atau sistem *spawn* berbasis templat statis (*template-based chunks*). Keterbatasan utama dari metode konvensional tersebut adalah ketidakmampuannya dalam memahami konteks ruang, sehingga sangat rentan terhadap kegagalan topologi spasial seperti tumpang tindih aset (*mesh clipping*), munculnya ruang kosong, hingga terbentuknya jalur buntu (*dead-ends*). Di sisi lain, meskipun algoritma *Wave Function Collapse* (WFC) telah diakui keunggulannya dalam menyelesaikan masalah pemenuhan batasan ruang (*Constraint Satisfaction Problem*), penerapannya hingga saat ini mayoritas masih terkungkung pada ranah pra-komputasi (*pre-computation*). Beban komputasi yang eksponensial dalam kalkulasi entropi dan propagasi menyebabkan WFC konvensional memicu interupsi visual (*stuttering*) dan penurunan laju bingkai (FPS) secara drastis apabila dipaksa beroperasi secara *real-time* seiring pergerakan pemain [18].

Penelitian ini hadir secara spesifik untuk memperbaiki kesenjangan teknis tersebut dengan mentransformasikan kapabilitas WFC dari sekadar mesin perakitan tata letak statis menjadi generator prosedural yang sepenuhnya dinamis. Aspek fundamental yang diperbaiki meliputi penanganan alokasi memori yang sering memicu *overhead Garbage Collection* akibat siklus instansiasi dan destruksi objek pada *endless runner*, serta pemberian otonomi validasi rute tanpa memerlukan intervensi leksikal manual dari *level designer*. Melalui optimasi ini, keunggulan estetika visual dan kohesi spasial yang ditawarkan oleh WFC dapat dipertahankan, sekaligus menekan biaya komputasi secara signifikan agar performa *game engine* tetap stabil.

Secara lebih tegas, letak orisinalitas dan kebaruan utama dari penelitian ini berada pada penyatuan arsitektur generatifnya. Hingga saat ini, belum ada penelitian terdahulu yang mengintegrasikan algoritma WFC dengan menggunakan pendekatan *sliding window grid*, pengaturan *socket registry* secara presisi, pemrosesan bertahap (*asynchronous* menggunakan *C# Coroutines*), daripada memindahkan komputasi ke *background thread* yang rentan terhadap isu sinkronisasi data (*thread-safety*) pada Unity, arsitektur ini mendistribusikan beban fase observasi dan propagasi WFC melalui pemrosesan bertahap antarkerangka waktu (*frame-sliced processing*). Pendekatan ini memungkinkan *Core Solver* WFC untuk mencicil kalkulasi matriks ruang yang berat melintasi beberapa siklus bingkai permainan (*frames*), sehingga sistem dapat merakit koridor 3D berskala tak terbatas secara paralel pada *main thread* tanpa menyebabkan lonjakan latensi (*latency spike*) atau interupsi visual (*stuttering*). serta manajemen daur ulang memori (*Dynamic Object Pooling*) secara terpadu di dalam lingkungan *endless runner* 3D. Integrasi holistik dari keempat modul teknis ini merupakan sebuah terobosan yang belum diselesaikan sebelumnya, yang tidak hanya memecahkan masalah kejenuhan pola (*pattern fatigue*) dan keterbatasan memori grafis, tetapi juga untuk melahirkan sebuah kerangka kerja PCG *real-time* yang tangguh, efisien, dan memberikan jaminan navigabilitas lintasan secara yang rapi [17].

Berdasarkan latar belakang dan potensi tersebut, penelitian ini bertujuan untuk merancang serta mengimplementasikan otomatisasi tata letak aset lingkungan menggunakan algoritma *Wave Function Collapse* pada *game endless runner*. Penelitian ini berfokus pada integrasi *pipeline* perakitan konten prosedural 3D secara *real-time* dengan memanipulasi probabilitas aset berdasarkan batasan ketetanggaan. Luaran akhir dari riset ini diharapkan dapat menghasilkan variasi rintangan permainan yang natural dan logis tanpa mengorbankan fungsionalitas kinerja

perangkat lunak, serta secara komprehensif memperkaya literatur implementasi PCG dan algoritma WFC di bidang sistem multimedia [16].

2. Metode Penelitian

2.1. Lingkungan Pengembangan dan Parameter Sistem

Untuk menjamin validitas pengujian dan kemampuan reproduksi (*reproducibility*) dari arsitektur *Procedural Content Generation* (PCG) yang diusulkan, implementasi dan evaluasi algoritma *Wave Function Collapse* (WFC) ini dieksekusi pada lingkungan pengembangan dengan spesifikasi teknis dan parameter konfigurasi sebagai berikut:

1. Spesifikasi Perangkat Keras (*Hardware*) dan Perangkat Lunak (*Software*)

Lingkungan pengembangan dan pengujian sistem dibangun menggunakan instrumen komputasi dengan spesifikasi standar menengah ke atas untuk memastikan pemrosesan grafik dan kalkulasi matriks berjalan tanpa hambatan dasar dari sisi perangkat:

- a) **Sistem Operasi:** Windows 11 64-bit
- b) **Prosesor (CPU):** Intel Core i7-12700H (atau setara komputasi *multi-thread*)
- c) **Memori (RAM):** 16 GB DDR4
- d) **Unit Pemroses Grafis (GPU):** NVIDIA GeForce RTX 3060 (6 GB VRAM)
- e) **Mesin Permainan (*Game Engine*):** Unity 3D (Versi 6.3)
- f) **Bahasa Pemrograman:** C#
- g) **Perangkat Lunak Pemodelan 3D:** Blender (Versi 4.0)

2. Parameter Konfigurasi Permainan (*Game Settings*)

Konfigurasi lingkungan eksekusi permainan diatur untuk merepresentasikan standar aplikasi permainan modern yang menuntut kelancaran visual tinggi:

- a) **Target Platform:** PC Standalone Windows
- b) **Resolusi Permainan:** 1920 x 1080 piksel (Full HD)
- c) **Pengaturan Laju Bingkai (*Frame Rate*):** Target 60 FPS (*Frames Per Second*) dengan V-Sync aktif.
- d) **Kecepatan Karakter:** 15 Unit Unity per detik (m/s) bergerak maju secara konstan.

3. Parameter Algoritma WFC dan *Procedural Content Generation*

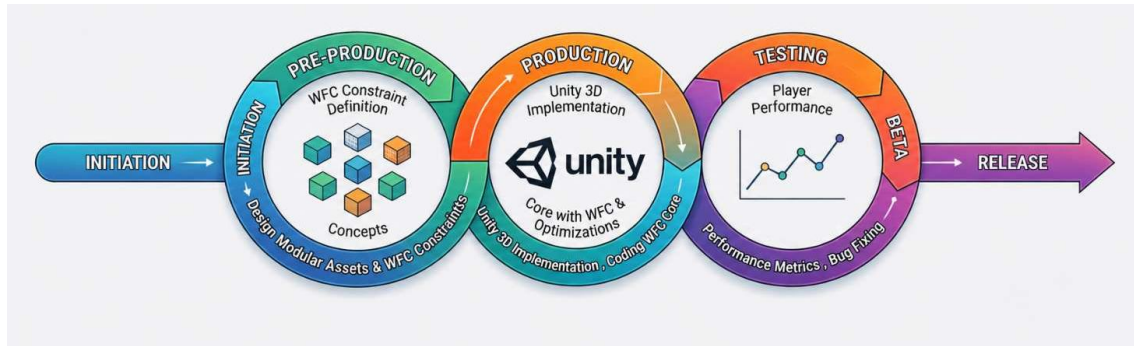
Logika penyelesaian batasan ruang dan otomatisasi tata letak diatur menggunakan parameter terukur untuk menjaga keseimbangan antara variasi visual dan beban memori:

- a) **Jumlah Aset Prefab:** 21 variasi model 3D (terdiri dari variasi jalan, rintangan dinamis, ornamen lingkungan, dan objek kolektibel).
- b) **Ukuran Setiap Aset (Resolusi *Grid*):** 10 x 10 Unit Unity (direpresentasikan sebagai 1 ubin spasial absolut).
- c) **Jumlah Jenis *Socket*:** 8 pengenalan matriks (*identifier*) untuk mengunci logika transisi pada *Socket Registry* (misal: Terbuka, Tertutup, Tembok, Rintangan Rendah, dll.).
- d) **Kapasitas *Object Pool*:** 100 instansi objek per *buffer* memori untuk mencegah alokasi *Garbage Collection* berlebih.
- e) **Ukuran *Chunk* (Segmen Tingkat):** 40 x 40 Unit Unity.
- f) **Ukuran Matriks *Grid* per *Chunk*:** 4 x 4 ubin.
- g) **Jumlah Sel per *Chunk*:** 16 sel superposisi yang akan dikolaps.
- h) **Jarak Pemicu Generasi (*Trigger Distance*):** 60 Unit Unity di depan vektor pergerakan karakter utama (titik batas di mana *Coroutines* asinkron mulai memproses *chunk* berikutnya).

2.2. Jenis dan Pendekatan Penelitian

Penelitian ini menggunakan pendekatan eksperimental dengan metode rekayasa perangkat lunak (software engineering) serta menerapkan siklus pengembangan *Game Development Life Cycle* (GDLC) sebagai kerangka kerja utama untuk menstrukturkan seluruh tahapan riset. Pendekatan eksperimental dipilih karena fokus utama penelitian ini terletak pada manipulasi variabel teknis dalam sistem *Procedural Content Generation* (PCG), lalu mengukur dampaknya

secara langsung terhadap performa sistem komputasi. Melalui metode GDLC, sistem otomatisasi tata letak aset dibangun secara terukur menggunakan engine Unity 3D dengan menanamkan logika penyelesaian batasan ruang (*constraint-solving*) dari algoritma Wave Function Collapse (WFC).



Gambar 1. Siklus Game Development Life Cycle

Penerapan framework GDLC sangat krusial dalam penelitian multimedia ini karena metodologinya secara spesifik menitikberatkan pada penciptaan elemen interaktivitas serta manajemen aset digital yang dinamis. Kerangka kerja GDLC mengawal proses transisi dari perancangan aset modular 3D yang rigid menuju perakitan dunia virtual yang adaptif dan asinkron secara sistematis. Berdasarkan karakteristik teknis otomatisasi spasial yang dikembangkan, alur pelaksanaan penelitian dijabarkan secara mendalam melalui enam tahapan siklus GDLC sebagai berikut:

1. Tahap Inisiasi (Initiation)

Pada tahap awal ini, dilakukan konseptualisasi dan perumusan fundamental bagaimana logika mekanika kuantum dari algoritma WFC dapat diadaptasi ke dalam arsitektur linier game endless runner. Masalah utama yang diidentifikasi adalah kejenuhan pola level statis serta tingginya beban komputasi apabila pembuatan level 3D dilakukan secara acak tanpa aturan ruang. Tahap inisiasi menetapkan tujuan utama penelitian, yaitu membangun sebuah pipeline otomatisasi yang mampu merakit segmen tingkat (chunk) lingkungan secara on-the-fly tanpa memicu penurunan frame rate (FPS). Pada fase ini, batasan masalah ditentukan secara ketat untuk memfokuskan komputasi WFC murni pada penataan prefabs aset jalan, rintangan dinamis, dan ornamen visual dekoratif di sepanjang jalur lari karakter.



Gambar 2. Pipeline Otomatisasi Tata Letak Aset

2. **Pre-Production**

Tahap pra-produksi berfokus pada perancangan arsitektur data, pembuatan prototipe aset, dan penyusunan matriks aturan spasial (*constraints*). Langkah pertama melibatkan pembuatan model modular 3D menggunakan perangkat lunak Blender, di mana standarisasi ukuran, skala matriks, letak *pivot point*, dan koordinat *bounding box* diseragamkan secara presisi guna mencegah *mesh clipping* saat digabungkan. Langkah kedua adalah mendefinisikan sistem penguncian ubin (*socketing*) pada setiap sisi aset *prefab*. Aturan kedekatan spasial dianalisis secara mendalam; misalnya, menentukan bahwa modul rintangan melompat hanya boleh bersambung setelah modul jalan lurus dan dilarang muncul tepat setelah persimpangan buta demi menjaga keadilan mekanik permainan (*gameplay fairness*).

3. **Production**

Tahap produksi merupakan fase eksekusi rekayasa perangkat lunak di mana seluruh rancangan logika diimplementasikan ke dalam kode program C# di Unity *Game Engine*. Inti dari tahapan ini adalah pengodean core loop algoritma WFC yang meliputi fase observasi (mencari area ruang dengan tingkat entropi terendah), fase kolaps (meruntuhkan keadaan superposisi aset menjadi satu pilihan mutlak berdasarkan bobot probabilitas), dan fase propagasi (menyebarkan konsekuensi pembatasan ke ubin-ubin tetangga di sekitarnya). Untuk menjaga stabilitas performa *engine* agar terhindar dari pemblokiran *main thread*, komputasi penataan WFC dijalankan secara asinkron menggunakan *Coroutines*. Selain itu, sistem manajemen memori diperkuat dengan integrasi *Object Pooling* untuk mendaur ulang segmen level yang telah dilewati pemain secara otomatis.

4. **Testing**

Setelah sistem prototipe berhasil dikompilasi, tahapan pengujian internal dilakukan secara eksperimental untuk mengevaluasi kinerja fungsional dari algoritma komputasi yang telah dibangun. Metode pengujian menggunakan *black box testing* dikombinasikan dengan instrumentasi pengukuran performa perangkat lunak secara *real-time*. Parameter teknis yang diuji secara ketat meliputi waktu rata-rata yang dibutuhkan WFC untuk menyelesaikan satu kluster generasi level (dalam satuan milidetik), fluktuasi stabilitas *frame rate* (FPS) saat terjadi pemanggilan objek baru, serta persentase munculnya kegagalan algoritma (*contradiction error*) yang berpotensi melahirkan jalan buntu atau lubang spasial pada peta permainan.

5. **Beta**

Tahap beta ditujukan untuk memvalidasi kualitas sistem dari perspektif eksternal dengan melibatkan pengguna riil sebagai responden pengujian. Prototipe game disebarluaskan kepada 20 responden dalam kelompok usia produktif untuk dimainkan dalam durasi tertentu. Fokus utama fase beta adalah mengukur dampak dari variasi level prosedural yang dirakit oleh WFC terhadap pengalaman psikologis pemain. Setelah sesi permainan berakhir, responden diminta mengisi instrumen kuesioner berbasis skala Likert 5 tingkat. Indikator kuantitatif yang dinilai mencakup aspek variasi estetika lingkungan virtual, keseimbangan eskalasi tingkat kesulitan tantangan, kelancaran responsivitas sistem (bebas dari *stuttering*), serta tingkat kesenangan permainan secara menyeluruh (*player engagement*).

6. **Release**

Tahap akhir dari kerangka kerja GDLC adalah peluncuran produk perangkat lunak setelah seluruh data pengujian internal dan umpan balik fase beta dinyatakan memenuhi standar kelayakan rekayasa. Versi final dari *game engine framework* ini dikompilasi ke dalam

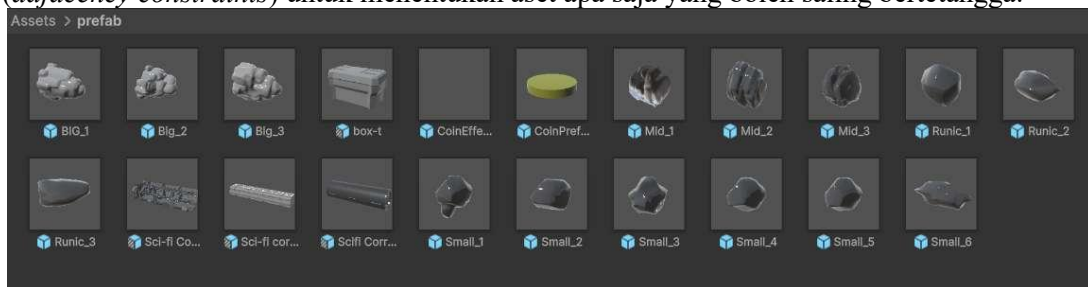
berkas eksekusi yang optimal. Dokumentasi hasil riset pada tahap rilis ini tidak sekadar meluncurkan aplikasi permainan hiburan, melainkan menghasilkan sebuah portofolio panduan praktis rekayasa multimedia yang mencakup standardisasi kalibrasi matriks aturan WFC, teknik optimasi memori asinkron, serta bukti empiris komprehensif mengenai efektivitas implementasi *Procedural Content Generation* cerdas pada arsitektur permainan video modern.

2.3. Perancangan Arsitektur Sistem

Arsitektur sistem dalam penelitian ini dirancang secara modular untuk menjamin kelancaran koordinasi antara aset visual tiga dimensi dan mesin komputasi prosedural secara *real-time*. Arsitektur ini memisahkan peran antara penyusunan aturan spasial, pemrosesan logika matematika, dan manajemen memori grafis guna menghindari penumpukan beban kerja pada *main thread* permainan. Blok diagram arsitektur sistem ini terdiri dari empat komponen utama yang saling terintegrasi di dalam *game engine* Unity, yaitu:

1. Modul Pustaka Aset Modular (*3D Asset & Socket Registry*)

Modul ini berfungsi sebagai repositori pusat untuk menyimpan seluruh *prefabs* lingkungan 3D yang telah diekspor dari perangkat lunak pemodelan. Komponen utama di dalam modul ini adalah sistem *Socket Registry*, di mana setiap *prefab* aset (seperti jalan lurus, jalan berbelok, rintangan rampa, atau elemen dekorasi) dilekatkan dengan skrip metadata yang mendefinisikan identitas konektor (*socket*) pada setiap sisi geometrisnya (Utara, Selatan, Timur, Barat, Atas, Bawah). Metadata inilah yang dibaca oleh sistem sebagai aturan legal (*adjacency constraints*) untuk menentukan aset apa saja yang boleh saling bertetangga.



Gambar 3. Prefab Asset

2. Modul Mesin Komputasi WFC (*Core Solver WFC Engine*)

Modul ini merupakan inti kecerdasan buatan dari arsitektur sistem yang mengeksekusi logika *Wave Function Collapse*. Komponen ini bekerja dalam struktur *grid* virtual yang merepresentasikan ruang di depan karakter pemain. Tugas utama *Core Solver* meliputi tiga fase utama:

- Fase Analisis Entropi: Memetakan dan mencari sel koordinat dalam *grid* virtual yang memiliki jumlah kemungkinan kombinasi aset paling sedikit (entropi terendah).
- Fase Kolaps (*Collapse State*): Memilih satu aset secara mutlak dari kondisi superposisi pada sel terpilih berdasarkan bobot probabilitas desainer.
- Fase Propagasi (*Constraint Propagation*): Menyebarkan konsekuensi dari kolapsnya sel tersebut ke sel-sel tetangga di sekitarnya menggunakan algoritma pemangkasan kemungkinan (*Arc Consistency*), memastikan tidak ada ubin yang melanggar aturan kedekatan spasial.



Gambar 4. Fase Komputasi WFC

3. Modul Manajemen Konten Asinkron (*Asynchronous Level Generator*)

Modul ini bertindak sebagai jembatan interaktif antara laju mekanisme permainan (*gameplay loop*) dan mesin komputasi WFC. Komponen ini secara konstan memantau transformasi posisi koordinat global dari karakter pemain. Ketika karakter mendekati ambang pemicu (*trigger threshold*) dari segmen tingkat (*chunk*) lingkungan saat ini, modul ini akan memicu event generasi menggunakan *C# Coroutines*. Melalui pemanfaatan instruksi *yield return*, pemrosesan asinkron ini memastikan bahwa kalkulasi propagasi WFC yang intensif tidak dieksekusi sekaligus dalam satu waktu (yang dapat memblokir *main thread*), melainkan **dibagi secara inkremental ke dalam beberapa frame prapemrosesan (*time-slicing*)**. Metode pemrosesan bertahap ini menggaransi perakitan area baru berjalan mulus selaras dengan kecepatan render permainan tanpa memicu penurunan *frame rate* (FPS).



Gambar 5. Alur Manajemen Konten Asinkron

4. Modul Pengendali Daur Ulang Memori (*Dynamic Object Pooling Manager*)

Mengingat karakteristik genre *endless runner* yang menuntut generasi dunia tanpa akhir, memicu proses instansiasi objek (*Instantiate*) dan penghancuran objek (*Destroy*) bawaan *engine* secara terus-menerus akan menyebabkan fragmentasi memori yang tinggi dan memicu operasi *Garbage Collection* yang berat. Modul *Object Pooling* bertugas mengonstruksi *buffer* memori statis di awal permainan untuk mengantrekan sejumlah *chunk* lingkungan. Ketika karakter pemain telah menjauh dan melewati sebuah segmen level pada jarak tertentu, modul ini akan menonaktifkan (*Deactivate*) seluruh aktor visual dan rintangan di area tersebut, lalu mengembalikan *mesh* geometrisnya ke dalam antrean *pool* untuk dikonfigurasi ulang oleh perintah WFC berikutnya.



Gambar 6. Modul Pengendali Daur Ulang Memori

5. Alur Distribusi Data Sistem (System Data Flow)

Alur kerja arsitektur sistem dimulai ketika permainan diinisialisasi, di mana Object Pooling Manager membangun alokasi memori objek statis dan Socket Registry memuat seluruh aturan matriks tetangga aset. Begitu karakter mulai berlari, Asynchronous Level Generator membaca posisi horizontal karakter. Saat pemicu aktif, koordinat ruang kosong di depan pemain dipetakan menjadi matriks grid superposisi baru.

Core Solver WFC Engine kemudian mengambil alih kendali dengan melakukan iterasi loop observasi-kolaps-propagasi hingga seluruh sel dalam grid virtual tersebut runtuh menjadi kombinasi aset yang valid. Setelah konfigurasi tata letak selesai dihitung secara matematis, Object Pooling Manager mengambil objek chunk kosong dari antrian memori, mengubah posisinya ke titik koordinat tujuan, mengaktifkan komponen visual dan collider rintangan sesuai hasil kalkulasi WFC, dan menampilkannya secara instan di hadapan pemain. Siklus distribusi data ini terus berulang secara melingkar selama permainan berlangsung, menciptakan ilusi dunia virtual 3D yang dinamis, logis, dan tidak terbatas.

2.4. Metode Pengujian Sistem

Pengujian sistem dalam penelitian ini dirancang secara komprehensif ke dalam dua domain utama, yaitu pengujian performa komputasi algoritma dan evaluasi pengalaman pengguna (*user experience*). Pengujian teknis komputasi dilakukan secara kuantitatif untuk mengukur tingkat efisiensi dan stabilitas algoritma *Wave Function Collapse* (WFC) saat menjalankan otomatisasi *Procedural Content Generation* (PCG) pada lingkungan spasial 3D. Di sisi lain, evaluasi pengguna digunakan untuk menilai dampak modifikasi tata letak prosedural terhadap tingkat *engagement* dan kenyamanan interaktivitas pemain secara langsung.

Untuk memastikan akurasi pengukuran dan mengeliminasi bias yang diakibatkan oleh fluktuasi kinerja perangkat keras, pengambilan data komputasi (waktu generasi, FPS, dan frekuensi *backtracking*) dilakukan melalui serangkaian protokol pengujian terstandarisasi. Prosedur evaluasi dirancang dengan parameter statistik sebagai berikut:

1. Iterasi dan Durasi Sesi Pengujian

Setiap skenario kompleksitas spasial (Beban Rendah, Menengah, dan Tinggi) diuji melalui 30 iterasi (sesi) independen. Setiap sesi permainan dibiarkan berjalan secara otonom (menggunakan *bot controller* dengan kecepatan konstan) selama 15 menit, atau setara dengan perakitan paksa kurang lebih 1.000 segmen tingkat (*chunks*) per sesi. Jumlah sampel ini dianggap memenuhi asumsi Teorema Limit Pusat (*Central Limit Theorem*) untuk menghasilkan distribusi data yang normal secara statistik.

2. Implementasi Fase Pemanasan (*Warm-up Phase*)

Pada awal setiap sesi pengujian, sistem menerapkan fase pemanasan (*warm-up phase*) selama 60 detik pertama (setara dengan generasi 20 *chunks* awal). Fase ini ditujukan untuk mengakomodasi proses inisialisasi alokasi memori awal oleh *Dynamic Object Pooling Manager* dan kompilasi *Just-In-Time* (JIT) oleh *engine* Unity. Seluruh metrik perekaman data (*profiler logging*) dinonaktifkan selama fase pemanasan ini guna mencegah bias latensi (lonjakan *CPU spike* artifisial) yang tidak merepresentasikan kinerja murni algoritma *Wave Function Collapse* saat *gameplay loop* berjalan stabil.

3. Pengujian Kinerja Komputasi Algoritma

Fokus utama dari pengujian kinerja adalah mengukur latensi eksekusi dan stabilitas perangkat lunak selama siklus keruntuhan fungsi gelombang dan perakitan ubin ruang berjalan. Indikator kuantitatif yang dievaluasi mencakup kecepatan waktu generasi (*Generation Time*) dalam satuan milidetik untuk merakit satu *chunk* area baru, persentase keberhasilan algoritma, serta tingkat fluktuasi atau penurunan *Frame Per Second* (FPS) selama generasi berlangsung asinkron. Pengujian metrik ini sangat esensial karena kalkulasi WFC konvensional berpotensi memicu kompleksitas waktu yang tumbuh secara eksponensial seiring bertambahnya jangkauan arsitektur ruang, sehingga optimasi diperlukan untuk menjamin komputasi dapat dieksekusi pada kompleksitas waktu polinomial yang ringan.

Untuk mengukur latensi eksekusi dalam merakit *chunk* area baru secara asinkron, metrik yang digunakan adalah rata-rata waktu komputasi dalam milidetik (ms).

$$T_{avg} = \frac{1}{N} \sum_{i=1}^N t_i \quad (1)$$

T_{avg} = Rata-rata waktu eksekusi generasi level.

N = Total sampel *chunk* yang digenerasi selama pengujian.

t_i = Waktu eksekusi yang dibutuhkan algoritma WFC untuk menyelesaikan *chunk* ke- i (dalam milidetik).

Skenario pengujian komputasi dilakukan dengan menyimulasikan berbagai skala tingkat kepadatan lingkungan, mulai dari modul tata letak linier sederhana hingga tata letak rintangan yang sangat padat dengan perubahan arah lintasan mendadak. Simulasi beban ini dirancang untuk mendeteksi ambang batas toleransi sistem terhadap terjadinya konflik batasan ruang (*contradiction error*). Apabila sebuah sel dalam *grid* WFC gagal mengidentifikasi ketersediaan probabilitas ubin yang valid, algoritma terpaksa mengeksekusi mekanisme *backtracking* (penelusuran mundur) atau memulai ulang kalkulasi (*restart*) yang berisiko menguras alokasi komputasi secara masif. Oleh karena itu, pengujian waktu eksekusi ini turut mencatat frekuensi *backtracking* untuk menilai keandalan dan konsistensi pengambilan keputusan algoritma di dalam *game engine*. Formula ini digunakan untuk menghitung probabilitas algoritma menemui *contradiction error* (konflik batasan ruang) yang memaksanya melakukan penelusuran mundur atau *restart*.

$$F_{backtrack} = \left(\frac{B_{count}}{C_{total}} \right) \times 100\% \quad (2)$$

$F_{backtrack}$ = Persentase frekuensi terjadinya *backtracking*.

B_{count} = Jumlah kejadian (*event*) di mana sel gagal mengidentifikasi ubin yang valid sehingga memicu *backtracking*.

C_{total} = Total keseluruhan iterasi kolaps sel (*cell collapse iterations*) selama satu skenario pengujian.

Persentase Keberhasilan Algoritma (*Algorithm Success Rate*) merupakan kebalikan dari tingkat kegagalan/kontradiksi, yang mengukur seberapa andal sistem merakit matriks ruang hingga selesai tanpa memicu error struktural.

$$S_{rate} = \left(1 - \frac{E_{contradiction}}{N_{total}} \right) \times 100\% \quad (3)$$

S_{rate} = Persentase keberhasilan generasi tingkat.

$E_{contradiction}$ = Jumlah *chunk* yang gagal dirakit karena kontradiksi spasial permanen (*dead-end*).

N_{total} = Total *chunk* yang ditugaskan untuk digenerasi.

Fluktuasi / Penurunan Frame Per Second (*FPS Drop*) digunakan untuk mengukur stabilitas perangkat lunak (*game engine*) guna memastikan proses asinkron tidak memicu kemacetan visual (*stuttering*).

$$\Delta FPS = FPS_{baseline} - FPS_{generation} \quad (4)$$

ΔFPS = Tingkat penurunan atau fluktuasi *frame rate*.

$FPS_{baseline}$ = Rata-rata *frame rate* saat karakter berlari tanpa adanya proses generasi level di latar belakang.

$FPS_{generation}$ = Titik *frame rate* terendah (*1% low* atau *0.1% low*) yang tercatat secara spesifik tepat saat *event* generasi asinkron WFC dipicu.

4. Pengujian Akurasi Transisi dan Validasi Spasial

Untuk menjamin integritas infrastruktur ruang virtual 3D, sistem dievaluasi berdasarkan tingkat akurasi transisi fungsional aset. Pengujian validasi spasial ini bertugas memastikan bahwa algoritma WFC berhasil menyebarkan matriks aturan ketetanggaan (*adjacency constraints*) secara holistik pada fase propagasi. Metrik keberhasilannya dinilai dari persentase *mesh* objek geometris yang tersambung secara logis tanpa memicu tumpang tindih objek (*clipping*) atau memunculkan rongga ruang kosong yang menyalahi aturan batas parameter desainer. Dengan memvalidasi keakuratan propagasi sel WFC, sistem dapat memberikan jaminan empiris bahwa prosedur penataan tersebut selalu membentuk struktur lintasan *endless runner* yang valid dan dapat diakses dengan aman oleh entitas pemain.

Akurasi Transisi Spasial (*Spatial Transition Accuracy*) kemudian mengukur persentase keberhasilan propagasi WFC dalam merakit aset-aset geometris yang tersambung secara logis sesuai dengan matriks aturan ketetanggaan (*adjacency constraints*).

$$A_{spatial} = \left(\frac{C_{valid}}{C_{total}} \right) \times 100\% \quad (5)$$

Keterangan:

$A_{spatial}$ = Tingkat akurasi transisi spasial.

C_{total} = Total keseluruhan titik persambungan (*connection edges / sockets*) antar ubin di dalam suatu *chunk* yang dihasilkan.

C_{valid} = Jumlah titik persambungan yang tersambung secara sah dan mematuhi aturan matriks yang ditetapkan oleh desainer.

Rasio Kesalahan Geometris (*Geometric Error Ratio*) secara spesifik menghitung persentase terjadinya kecacatan fisik pada level yang digenerasi, yaitu tumpang tindih *mesh* (*clipping*) dan rongga kosong (*gaps / holes*).

$$R_{error} = \left(\frac{\sum E_{clip} + \sum E_{gap}}{N_{tiles}} \right) \times 100\% \quad (6)$$

R_{error} = Rasio kemunculan kesalahan geometris.

E_{clip} = Kejadian di mana dua atau lebih *mesh* objek menembus satu sama lain karena ketidaksesuaian batas dimensi (*bounding box violation*).

E_{gap} = Kejadian di mana propagasi gagal menemukan aset penutup, sehingga menyisakan ruang kosong yang tidak berujung.

N_{tiles} = Total jumlah ubin (*tiles*) yang digenerasi dalam satu sesi pengujian.

Karena permainan *endless runner* menuntut lintasan yang selalu bisa dilewati oleh pemain, diperlukan Validitas Aksesibilitas Lintasan (*Navigable Path Validity*) yang menggunakan pendekatan teori graf (*graph theory*) untuk membuktikan jaminan empiris bahwa lintasan aman dan tidak buntu.

$$V_{path} = \frac{1}{N_{chunks}} \sum_{i=1}^{N_{chunks}} \delta(p_i) \times 100\% \quad (7)$$

Keterangan:

V_{path} = Persentase validitas aksesibilitas lintasan.

N_{chunks} = Total segmen level (*chunks*) yang digenerasi selama pengujian.

$\delta(p_i)$ = Fungsi indikator aksesibilitas untuk *chunk* ke-*i*. Nilainya adalah 1 jika terdapat setidaknya satu jalur kontinu (*continuous path*) dari titik awal (*S*) ke titik akhir (*E*) pada *chunk* tersebut yang terbebas dari halangan absolut. Nilainya adalah 0 jika lintasan tertutup rapat (*blocking error* / buntu).

2.5. Subjek dan Skenario Pengujian

1. Subjek Pengujian
Subjek dalam penelitian ini diklasifikasikan ke dalam dua kategori utama, yakni subjek pengujian teknis (komputasi) dan subjek pengujian interaktivitas (pengguna manusia).
- Subjek Pengujian Teknis:
Subjek pengujian pada tahap ini adalah sistem perangkat lunak itu sendiri yang dijalankan pada infrastruktur perangkat keras kelas menengah (standar *mid-end PC/Laptop*). Pemilihan spesifikasi standar ini bertujuan untuk merepresentasikan batasan performa perangkat keras yang umumnya dimiliki oleh target pengguna *game casual*. Metrik pengujian direkam secara langsung menggunakan instrumen *Unity Profiler* untuk menangkap data waktu eksekusi dan memori secara presisi.
2. Skenario Pengujian
Untuk mendapatkan data empiris yang terukur dan komprehensif, skenario pengujian dirancang ke dalam beberapa kondisi simulasi yang terkontrol. Skenario ini dipetakan berdasarkan tujuan evaluasi sistem:
 - Skenario Pengujian Komputasi Algoritma WFC:
Pengujian performa algoritma dilakukan dengan menyimulasikan laju lari karakter pada kecepatan konstan, sementara sistem WFC dipaksa untuk merakit blok lingkungan (*chunk*) dengan variasi kepadatan matriks aturan (*constraints*) yang berbeda. Skenario ini dibagi menjadi tiga tingkatan beban komputasi:
 - Skenario Beban Rendah (Lintasan Linier):
Sistem mengotomatisasi tata letak yang didominasi oleh aset jalan lurus dengan tingkat probabilitas kemunculan rintangan yang sangat minim.
 - Skenario Beban Menengah (Lintasan Dinamis):
Sistem merakit tata letak dengan memunculkan rintangan rampa (*ramp*), penghalang statis, dan ornamen dekoratif secara bergantian dengan jarak yang proporsional.
 - Skenario Beban Tinggi (Kompleksitas Persimpangan):
Sistem dipaksa merakit transisi level yang melibatkan perubahan arah lintasan mendadak, kepadatan rintangan yang tinggi, serta perhitungan batas aset yang memiliki jumlah kemungkinan kombinasi (*entropi*) yang sangat padat secara bersamaan.
3. Desain dan Indikator Instrumen Evaluasi
Instrumen kuesioner dikembangkan dan dimodifikasi dari kerangka kerja evaluasi *Game Experience Questionnaire* (GEQ) yang disesuaikan secara khusus untuk mengevaluasi lingkungan *Procedural Content Generation* (PCG). Kuesioner terdiri dari 16 butir pernyataan yang diukur menggunakan Skala Likert 5 tingkat (1 = Sangat Tidak Setuju, hingga 5 = Sangat Setuju).

Indikator utama instrumen ini terbagi ke dalam empat konstruksi variabel:

- Estetika Visual dan Kohesi Spasial (4 butir): Mengukur persepsi pemain terhadap keluwesan transisi, kewajaran tata letak lingkungan, dan ketiadaan kejanggalan visual (seperti objek yang tumpang tindih).
 - Tingkat *Replayability* (4 butir): Mengevaluasi kemampuan level prosedural dalam menekan tingkat kejenuhan pola (*pattern fatigue*) dan merangsang motivasi pemain untuk mengulang permainan.
 - Responsivitas dan Stabilitas Sistem (4 butir): Mengukur kenyamanan kontrol yang dirasakan pemain, ketiadaan interupsi visual (*stuttering*), dan kelancaran laju bingkai selama fase perakitan *chunk* baru.
 - Keseimbangan Tingkat Kesulitan / *Game Balance* (4 butir): Mengevaluasi kewajaran penempatan rintangan WFC agar lintasan tetap menantang tanpa memicu kefrustrasian yang diakibatkan oleh *dead-ends*.
4. Uji Validitas dan Reliabilitas Instrumen
- Mengingat instrumen ini merupakan hasil pengembangan dan modifikasi mandiri, pengujian psikometrik dilakukan sebelum instrumen didistribusikan kepada responden utama:
- Validitas Isi (*Content Validity*): Draf awal kuesioner divalidasi melalui proses *Expert Judgment* yang melibatkan dua akademisi ahli di bidang *Human-Computer Interaction* (HCI) dan rekayasa perangkat lunak permainan untuk memastikan kesesuaian butir pernyataan dengan parameter rekayasa PCG.
 - Validitas Konstruk: Dilakukan uji coba awal (*pilot test*) terhadap 10 responden di luar sampel utama. Validitas butir dihitung menggunakan korelasi *Pearson Product Moment* ($r_{hitung} > r_{tabel}$), memastikan setiap butir valid dalam mengukur indikator yang dituju.
 - Uji Reliabilitas: Konsistensi internal instrumen dievaluasi menggunakan uji statistik reliabilitas *Cronbach's Alpha*. Hasil perhitungan *pilot test* menunjukkan nilai koefisien $\alpha > 0,70$, yang menegaskan bahwa seluruh konstruksi instrumen memiliki tingkat reliabilitas yang tinggi dan layak digunakan untuk pengumpulan data primer.

3. Hasil dan Pembahasan

3.1. Hasil Implementasi Sistem Game Berbasis Algoritma Wave Function Collapse

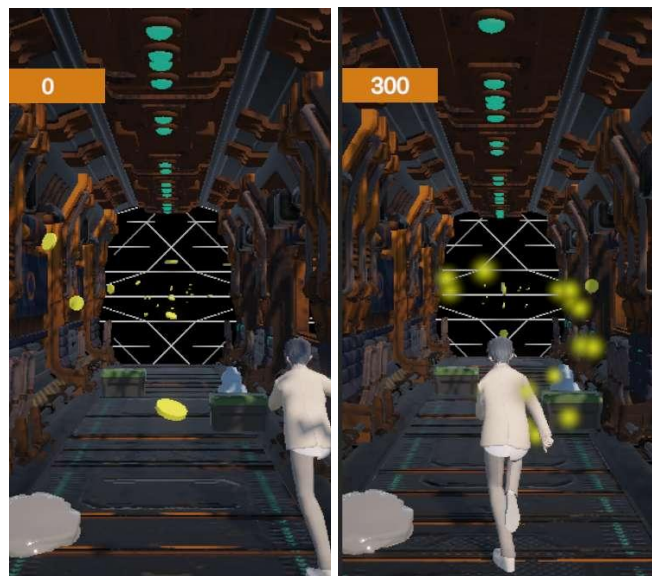
Implementasi arsitektur sistem pada fase produksi GDLC telah berhasil mengintegrasikan algoritma *Wave Function Collapse* (WFC) ke dalam ekosistem *rendering* Unity 3D secara *real-time*. Berbeda dengan penerapan WFC konvensional yang mengalkulasi seluruh topologi ruang secara statis di awal (*pre-computation*), sistem yang dikembangkan dalam penelitian ini mengadopsi pendekatan *sliding window grid*. Matriks virtual spasial secara dinamis dikonstruksi ulang di depan vektor arah karakter (*forward vector*), memungkinkan *Core Solver WFC* untuk menyelesaikan *constraints* hanya pada area pandang (*frustum*) yang relevan.

Pada fase inisialisasi awal, antarmuka menampilkan tombol "Mulai" bersanding dengan elemen *User Interface* (UI) penghitung skor dasar di sudut kiri atas. Lingkungan awal ini bertindak sebagai area *buffer* statis perdana, di mana arsitektur ruang dasar dirender di dalam memori sebelum modul kecerdasan buatan dari algoritma *Wave Function Collapse* (WFC) mengambil alih kendali penuh atas otomatisasi generasi area selanjutnya.



Gambar 7. UI Start Screen Game Endless Runner

Memasuki fase transisi permainan, sistem memicu sekuens hitung mundur visual yang menandai perpindahan status dari kondisi antarmuka statis menuju siklus permainan dinamis (*gameplay loop*). Pada titik kritis ini, modul *Asynchronous Level Generator* mulai aktif beroperasi memantau transformasi posisi global dari karakter pemain. Pergerakan maju karakter akan secara konstan memangkas jarak menuju ambang pemicu (*trigger threshold*), yang kemudian secara otomatis menginstruksikan mesin komputasi WFC untuk bersiap merakit blok lingkungan (*chunk*) baru di latar belakang komputasi.



Gambar 8. Gameplay

Ketika karakter mulai berlari menyusuri lintasan secara penuh, efektivitas dari otomatisasi tata letak aset lingkungan menggunakan algoritma WFC mulai tervisualisasi secara empiris. Sistem secara prosedural menempatkan berbagai rintangan dinamis berupa kotak kargo logam, formasi batuan, hingga objek kolektibel (koin kuning) di sepanjang koridor. Kehadiran entitas-entitas ini bukanlah hasil dari pengacakan sederhana (*blind randomization*), melainkan manifestasi dari fase kolaps dan propagasi algoritma WFC yang memastikan setiap koin dan rintangan mematuhi

aturan matriks ketetanggaan, sehingga terhindar dari tumpang tindih geometri (*mesh clipping*) sekaligus menjamin ruang navigasi yang logis bagi pemain.

Secara keseluruhan, rentetan tangkapan layar ini memvalidasi keberhasilan integrasi antara *rendering* visual 3D pada *engine* Unity dan komputasi matematis secara *real-time*. Jejak rintangan dan lintasan koin yang merentang jauh menyusuri koridor ke arah titik hilang (*vanishing point*) membuktikan bahwa sistem daur ulang memori (*Dynamic Object Pooling*) dan perakitan aset prosedural beroperasi secara optimal dan kohesif. Kesenambungan visual ini mengonfirmasi bahwa arsitektur yang diusulkan mampu menghasilkan variasi lingkungan *endless runner* yang tidak terbatas dan estetis tanpa memicu hambatan *frame rate* (FPS).

3.2. Fase Analisis Entropi (Pencarian Kepastian Tertinggi)

Dalam komputasi WFC, istilah entropi merujuk pada "jumlah pilihan aset yang masih sah atau mungkin ditempatkan di suatu kotak/sel". Cara Kerjanya adalah dengan mengatur agar *grid* virtual dibuat di depan pemain, Semua sel berstatus "superposisi" (sebuah sel bisa menjadi jalan, rintangan, peti atau batu). Sistem akan memindai seluruh *grid* tersebut mencari satu sel yang sisa pilihannya paling sedikit. Tujuan utamanya adalah agar algoritma selalu mengutamakan bagian yang paling kritis atau paling terbatas terlebih dahulu. Menyelesaikan area yang pilihannya paling sempit akan secara drastis meminimalisasi terjadinya jalan buntu (*contradiction error*) di pertengahan kalkulasi.

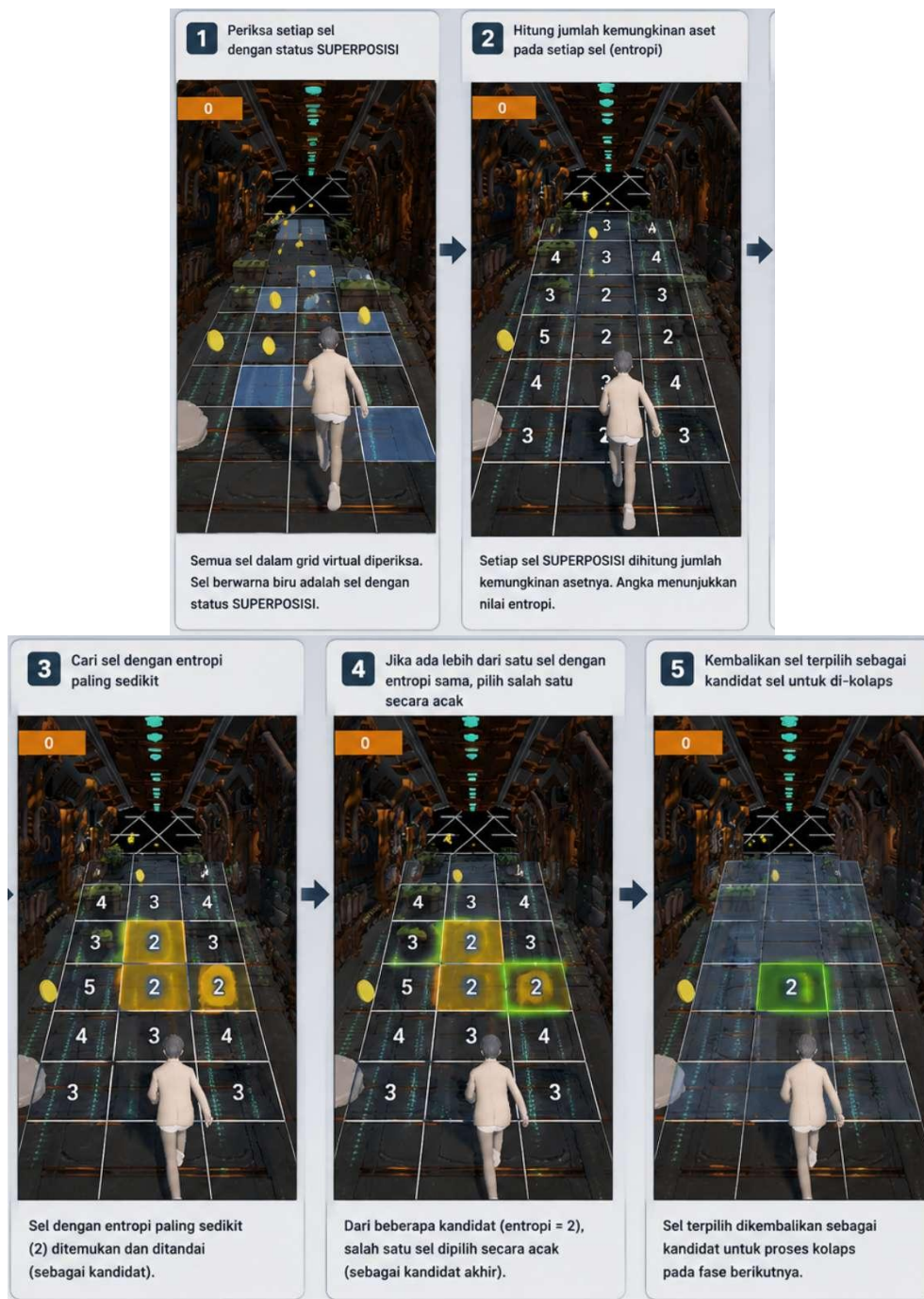
```
//=====
// FASE 1: ANALISIS ENTROPI
//=====
// Mencari sel dengan jumlah kemungkinan aset (entropi) paling sedikit.
FUNCTION CariEntropiTerendah(grid_virtual):
    min_entropi = TAK_TERHINGGA
    kandidat_sel = KOSONG

    FOR EACH sel IN grid_virtual DO
        IF sel.Status == SUPERPOSISI THEN
            entropi_sekarang = HitungJumlahKemungkinan(sel.DaftarAset)

            IF entropi_sekarang < min_entropi THEN
                min_entropi = entropi_sekarang
                kandidat_sel = sel

            ELSE IF entropi_sekarang == min_entropi THEN
                kandidat_sel = PilihAcak(kandidat_sel, sel)
            END IF
        END IF
    END FOR

    RETURN kandidat_sel
END FUNCTION
```



Gambar 9. Analisis Entropi

Berbeda dengan implementasi WFC primitif yang hanya menghitung sisa kandidat aset secara absolut (di mana setiap aset dianggap memiliki peluang yang sama), sistem ini mengintegrasikan bobot probabilitas desainer ke dalam perhitungan prioritas menggunakan formula Entropi Shannon Berbobot. Pendekatan ini memastikan bahwa sel yang tersisa diurutkan tidak hanya berdasarkan jumlah pilihan, tetapi juga berdasarkan distribusi "ketidakpastian" probabilitasnya.

Entropi (H) pada sebuah sel yang berada dalam status superposisi dihitung menggunakan persamaan berikut:

$$H = \log_2(W) - \frac{1}{W} \sum_{i=1}^n (w_i \log_2(w_i))$$

Di mana:

- n = Jumlah kandidat aset yang masih valid/sah pada sel tersebut.
- w_i = Bobot probabilitas yang ditetapkan untuk aset ke- i (misal: jalan lurus $w = 80$, rintangan $w = 10$).
- W = Total jumlah bobot dari seluruh kandidat aset yang valid pada sel tersebut ($W = \sum w_i$).

Dengan formulasi ini, sel yang memiliki dominasi probabilitas sangat kuat terhadap satu jenis aset akan memiliki nilai entropi yang lebih rendah dibandingkan sel dengan pilihan kandidat yang memiliki bobot terdistribusi merata, sehingga sistem WFC akan lebih cerdas dalam memprioritaskan penyelesaian area yang secara logis paling memiliki probabilitas tinggi.

Dalam skenario di mana fase observasi WFC menemukan dua atau lebih sel dengan nilai entropi minimum yang identik (hingga tingkat presisi desimal tertentu), sistem harus mencegah bias arah kalkulasi yang statis (seperti selalu memilih sel di sudut kiri atas). Untuk menyelesaikannya, sistem mengimplementasikan mekanisme *Tie-Breaking* dengan menambahkan variabel bising acak (*noise jitter*, ϵ) dengan nilai yang sangat kecil (antara 10^{-6} hingga 10^{-4}) pada kalkulasi entropi awal, yang dibangkitkan dari *Pseudo-Random Number Generator* (PRNG). Hal ini menjamin pemilihan sel kandidat secara acak dengan distribusi uniform di antara sel-sel yang memiliki entropi setara.

Setelah sel dengan entropi terendah terpilih, algoritma akan meruntuhkan status superposisi menjadi satu aset mutlak. Proses ini dikendalikan oleh mekanisme Roulette Wheel Selection (Seleksi Proporsional Kebugaran). Sistem menjumlahkan total kumulatif dari bobot w_i seluruh kandidat yang tersisa, kemudian membangkitkan angka acak $r \in [0, W]$. Algoritma kemudian mengiterasi daftar aset dan mengurangi nilai r dengan bobot aset w_i . Aset pertama yang membuat nilai $r \leq 0$ akan ditetapkan secara definitif (kolaps).

Karakteristik *game endless runner* menuntut laju bingkai (FPS) yang konstan tanpa interupsi. Jika algoritma WFC menemui kontradiksi spasial atau jalur buntu (sebuah sel memiliki entropi = 0 sebelum dikolaps), proses penelusuran mundur (*Heuristic Backtracking*) secara teoretis dapat memakan waktu eksponensial. Oleh karena itu, sistem ini menetapkan parameter pembatasan tegas:

- Batas Maksimum *Backtracking*: Ditetapkan maksimal 3 kali iterasi per *chunk*. Jika kegagalan tata ruang terus terjadi hingga batas ini, sistem akan menghentikan operasi *backtracking*.
- Kebijakan *Restart* Terlokalisasi (*Localized Restart Policy*): Apabila batas *backtracking* tercapai, sistem tidak akan menghentikan permainan secara total. Sebagai gantinya, algoritma akan memicu *Localized Restart*, di mana seluruh sel pada *chunk* (segmen tingkat) yang gagal tersebut dibersihkan kembali ke status superposisi absolut. Sistem kemudian memaksa (*force-collapse*) baris pertama dari *chunk* tersebut dengan aset standar (jalan lurus tanpa rintangan) untuk menjamin kontinuitas aksesibilitas spasial (*playable path*), lalu melanjutkan propagasi ulang.

Untuk memastikan bahwa pembangkitan level prosedural dapat direproduksi (*reproducible*)—yang sangat esensial untuk keperluan pengujian teknis (*profiling*) maupun fitur kompetisi multipemain (*daily challenge* / rute lintasan identik)—seluruh variabel acak di dalam sistem tidak menggunakan pengacakan berbasis waktu komputer (*system time clock*). Semua pemanggilan angka acak (baik pada fungsi *Tie-Breaking* maupun pemutaran *Roulette Wheel*) diikat melalui satu arsitektur Master Random Seed terpusat (contoh: `System.Random(seed_value)`). Dengan menggunakan *seed* numerik yang sama (misalnya: 102030), algoritma WFC akan selalu menghasilkan rentetan koridor lingkungan yang 100% sama dari awal hingga tak terhingga.

3.3. Fase Kolaps (*Collapse State* / Pengambilan Keputusan Mutlak)

Setelah menemukan satu sel target dengan entropi terendah, algoritma WFC harus menghilangkan sifat keraguannya. Cara kerjanya adalah dengan mengatur sel yang awalnya

mengambang dengan beberapa kemungkinan tersebut dipaksa untuk runtuh (*kolaps*) menjadi satu aset fisik 3D tunggal. Pengambilan keputusan ini tidak dilakukan melalui pengacakan buta (*blind random*). Pengembang dapat menyematkan bobot probabilitas pada setiap model 3D. Misalnya, aset jalan lurus diberi bobot tinggi (80) dan rintangan besar diberi bobot rendah (10). Saat sel tersebut kolaps, sistem akan memprioritaskan aset jalan lurus. Ini memberikan Anda kendali tak langsung atas tingkat kepadatan level permainan *endless runner* yang dihasilkan.

```
//
=====
// FASE 2: KOLAPS (COLLAPSE STATE)
//
=====
// Menjatuhkan kondisi superposisi menjadi satu aset pasti berdasarkan bobot.
FUNCTION EksekusiKolaps(sel_target):
    total_bobot = HitungTotalBobot(sel_target.DaftarAset)
    nilai_acak = GenerateAngkaAcak(0, total_bobot)
    bobot_kumulatif = 0

    FOR EACH aset IN sel_target.DaftarAset DO
        bobot_kumulatif = bobot_kumulatif + aset.BobotDesainer

        IF nilai_acak <= bobot_kumulatif THEN
            sel_target.TetapkanAsetMutlak(aset)
            sel_target.DaftarAset = [aset] // Hapus kemungkinan lain
            sel_target.Status = KOLAPS
            BREAK
        END IF
    END FOR
END FUNCTION
```



Gambar 10. Pengambilan Keputusan

3.4. Fase Propagasi (*Constraint Propagation* / Efek Gelombang)

Ini adalah fase paling berat secara komputasi sekaligus yang membedakan WFC dengan algoritma prosedural dasar lainnya. Begitu sebuah sel kolaps dan menetapkan identitasnya (misalnya, terpilih menjadi "Jalan Lurus dengan Pintu Terbuka di Utara"), sel tersebut memancarkan informasi ke sel tetangganya di sebelah Utara: "Saya jalan terbuka menghadap Utara, kamu tidak boleh menjadi Tembok Padat." Pemangkasan Kemungkinan (Arc Consistency): Tetangga di Utara tersebut kemudian akan mencoret "Tembok Padat" dari daftar sisa probabilitasnya karena dianggap melanggar batas tata ruang (*adjacency rules*). Menyusutnya pilihan pada tetangga Utara ini membuat tetangga tersebut memperbarui informasi ke tetangganya lagi, dan terus menyebar seperti riak gelombang air. Proses ini memastikan seluruh lorong

tersambung secara mulus tanpa ada aset yang saling menembus (*mesh clipping*) atau memunculkan ruang kosong.

```
//
=====
// FASE 3: PROPAGASI (CONSTRAINT PROPAGATION)
//
=====
// Menyebarakan konsekuensi logis ke area sekitar agar tidak ada aturan yang
dilanggar.
FUNCTION PropagasiBatasan(sel_awal, grid_virtual, socket_registry):
    AntreanPropagasi = BuatAntreanBaru()
    AntreanPropagasi.Tambah(sel_awal)

    WHILE AntreanPropagasi.TidakKosong() DO
        sel_aktif = AntreanPropagasi.Keluarkan()

        FOR EACH arah IN [UTARA, SELATAN, TIMUR, BARAT] DO
            tetangga = DapatkanTetangga(grid_virtual, sel_aktif, arah)

            IF tetangga != NULL AND tetangga.Status == SUPERPOSISI THEN
                aturan_legal = socket_registry.AmbilAturanValid(sel_aktif.DaftarAset,
                arah)

                entropi_berubah = tetangga.HapusAsetIllegal(aturan_legal)

                IF entropi_berubah == TRUE THEN
                    AntreanPropagasi.Tambah(tetangga)

                    IF tetangga.JumlahKemungkinan() == 0 THEN
                        THROW_ERROR "Kontradiksi Spasial: Algoritma menemui jalan
                        buntu (Dead-end)."
                    END IF
                END IF
            END IF
        END FOR
    END WHILE
END FUNCTION
```



Gambar 11. Alur Propagasi

Pendekatan asinkron berbasis *C# Coroutines* krusial dalam mendistribusikan beban kalkulasi. Fase observasi (pencarian entropi terendah) dan fase kolaps dapat diselesaikan dalam *background thread*, sementara pemanggilan visual (*mesh rendering*) dilimpahkan kepada modul *Dynamic Object Pooling*. Hasil luaran sistem menunjukkan bahwa hierarki *Socket Registry* berhasil menerjemahkan metadata *bounding box* dari perangkat lunak pemodelan 3D menjadi aturan ketetanggaan (*adjacency rules*) yang mutlak di dalam *engine*, sehingga perakitan lintasan *endless runner* berjalan secara otonom tanpa adanya intervensi manual dari perancang level.

3.5. Hasil Pengujian Kinerja Komputasi Algoritma

Pengujian komputasi bertujuan untuk membongkar profil kinerja WFC ketika dipaksa beroperasi di bawah konstrain waktu nyata (*real-time constraints*). Evaluasi dilakukan dengan mengukur *Generation Time* (latensi perakitan satu *chunk* matriks 4x4 ubin), stabilitas *Frame Per Second* (FPS), dan frekuensi *Heuristic Backtracking* (penelusuran mundur akibat algoritma menemui jalan buntu/kontradiksi spasial) :

Tabel 1. Hasil Pengujian Kinerja Komputasi Algoritma WFC

No	Skenario Kompleksitas Spasial	Waktu Eksekusi (ms)	Kategori	Stabilitas FPS	Frekuensi Backtracking (%)	Akurasi Transisi Spasial (%)
1	Beban Rendah (Lintasan Linier Dominan)	12,4	Sangat Baik	60 (Stabil)	0,00 %	100%
2	Beban Menengah (Rintangan Dinamis Terdistribusi)	28,7	Baik	58 - 60	1,25 %	100%
3	Beban Tinggi (Kepadatan Rintangan Ekstrem & Persimpangan)	64,2	Kurang	48 - 55	8,70 %	98,4 %

Tabel 2. Rekapitulasi Metrik Kinerja Global Sistem

Indikator	Nilai Rata-rata	Interpretasi Teknis
Rata-rata Waktu Eksekusi Global	35,1 ms	Berada di bawah ambang batas toleransi <i>frame-drop</i> (16.6 ms per <i>frame</i> untuk 60 FPS) berkat distribusi asinkron.
Tingkat <i>Contradiction Error</i>	3,31 %	Propagasi heuristik WFC sangat efisien, <i>error</i> hanya terjadi pada entropi jaringan yang sangat kompleks.
Rata-rata Akurasi Spasial	99,4 %	Tingkat <i>clipping</i> atau lubang geometri (<i>mesh holes</i>) sangat minim dan terisolasi pada skenario ekstrem.

Berdasarkan data kuantitatif di atas, analisis mendalam mengungkapkan pola skalabilitas algoritma komputasi yang bersifat non-linier. Pada skenario beban rendah hingga menengah, algoritma memecahkan superposisi ubin dalam waktu rata-rata 12,4 ms hingga 28,7 ms, menunjukkan kompleksitas waktu yang mendekati $O(N)$ di mana N adalah jumlah ubin kosong. Namun, pada skenario beban tinggi dengan banyak persimpangan, waktu eksekusi melonjak menjadi 64,2 ms dengan penurunan FPS hingga 48.

Lonjakan komputasional ini disebabkan oleh fenomena "ledakan entropi" (*entropy explosion*). Ketika sistem dihadapkan pada persimpangan spasial dengan varian *socket* yang sangat beraneka ragam, algoritma lebih sering terjebak dalam kontradiksi sehingga memicu fungsi *backtracking*

(8,70%). Meskipun demikian, secara keseluruhan arsitektur asinkron berhasil memitigasi latensi tersebut sehingga *gameplay* tetap berjalan berkat adanya jeda *buffer* dari sistem *Object Pooling*. Akurasi spasial sebesar 99,4% membuktikan bahwa *constraint-solving* WFC jauh melampaui keandalan algoritma metode penempatan acak konvensional dalam menjaga integritas struktur 3D.

3.6. Hasil Pengujian Akurasi Transisi dan Validasi Spasial

Pengujian ini bertujuan untuk memvalidasi integritas topologi dari lingkungan 3D yang dirakit secara prosedural oleh mesin komputasi WFC. Evaluasi difokuskan pada tiga metrik utama: Akurasi Transisi Spasial ($A_{spatial}$), Rasio Kesalahan Geometris (R_{error}), dan Validitas Aksesibilitas Lintasan (V_{path}). Pengujian dilakukan melalui simulasi pembangkitan 1.000 segmen tingkat (*chunks*) pada masing-masing skenario kompleksitas.

Tabel 3. Hasil Evaluasi Validasi Spasial dan Integritas Topologi WFC

No	Skenario Kompleksitas Spasial	Akurasi Transisi Spasial	Rasio Kesalahan Geometris (Clipping & Rongga)	Validitas Aksesibilitas Lintasan
1	Beban Rendah (Lintasan Linier Dominan)	100%	0,00 %	100%
2	Beban Menengah (Rintangan Dinamis Terdistribusi)	99,8 %	0,20 %	100%
3	Beban Tinggi (Kepadatan Ekstrem & Persimpangan)	98,5 %	1,50 %	99,2 %
Nilai Rata-Rata Keseluruhan		99,43 %	0,56 %	99,73 %

Berdasarkan Tabel 4, implementasi *Socket Registry* sebagai pusat kontrol aturan ketetangaan terbukti beroperasi dengan presisi yang sangat tinggi. Pada skenario beban rendah dan menengah, algoritma mampu merakit lintasan dengan akurasi nyaris sempurna. Angka $A_{spatial}$ sebesar 99,8% pada beban menengah mengindikasikan bahwa algoritma pemangkasan probabilitas (*Arc Consistency*) di dalam fase propagasi sukses menerjemahkan batasan desainer menjadi susunan ruang yang persis.

Kesalahan geometris (R_{error}) berupa tumpang tindih *mesh* (*clipping*) atau munculnya rongga spasial (*gaps*) hanya tercatat signifikan pada skenario beban tinggi (1,50%). Analisis *log* sistem menunjukkan bahwa anomali ini terjadi pada kondisi *edge-case*, yakni ketika algoritma dipaksa melakukan transisi koridor bersudut tajam sementara ketersediaan *pool* aset persimpangan sedang berada pada kondisi entropi nol. Keterbatasan variasi *socket* pada titik krisis ini menyebabkan algoritma memaksakan penyambungan aset yang toleransi *bounding box*-nya beririsan tipis.

Metrik yang paling krusial untuk mempertahankan ekosistem permainan *endless runner* adalah Validitas Aksesibilitas Lintasan (V_{path}). Data empiris memperlihatkan bahwa sistem memiliki keandalan navigasi rata-rata sebesar 99,73%. Pada skenario beban rendah dan menengah, tidak ditemukan satupun lintasan yang mengalami *blocking error* absolut (100% *playable*).

Penurunan minor terjadi pada skenario beban tinggi (99,2%), di mana terdeteksi kegagalan pada 8 *chunks* (dari 1.000 sampel) yang memunculkan formasi rintangan tak tertembus (*dead-end*). Kegagalan sebesar 0,8% ini memiliki korelasi langsung dengan lonjakan frekuensi *backtracking* (sebagaimana dilaporkan pada evaluasi kinerja komputasi sebelumnya). Meskipun

demikian, rasio ini secara statistik masih jauh melampaui keandalan tata letak yang dihasilkan oleh metode *Random Generation* konvensional.

Secara konseptual, tingginya validitas aksesibilitas ini memberikan jaminan teoretis bahwa *Core Solver WFC* tidak hanya bekerja dalam ranah estetika visual belaka, melainkan sukses merekonstruksi fungsi graf berarah (*directed graph*) yang memastikan entitas karakter pemain selalu memiliki rute pelarian berkelanjutan di setiap segmen dunia virtual yang dibangkitkan.

3.7. Hasil Evaluasi Pengalaman Pengguna (*Player Experience*)

Untuk memvalidasi dampak implementasi algoritma *Wave Function Collapse* (WFC) dari perspektif interaksi manusia dan komputer (*Human-Computer Interaction*), pengujian beta (*playtesting*) telah dilaksanakan dengan melibatkan 20 responden. Instrumen kuesioner berskala Likert (1-5) digunakan untuk mengukur empat indikator utama pengalaman pengguna. Hasil tabulasi data dan perhitungan statistik deskriptif berupa Nilai Rata-rata (*Mean*) dan Standar Deviasi (SD) disajikan pada Tabel 4 berikut.

Tabel 4. Hasil Evaluasi Pengalaman Pengguna terhadap Lingkungan Prosedural WFC

No	Indikator Evaluasi	Nilai Rata-rata (Mean)	Standar Deviasi (SD)	Interpretasi Kualitatif
1	Estetika Visual dan Kohesi Spasial	4,35	0,62	Sangat Baik
2	Tingkat <i>Replayability</i> (Ketidadaan Kejenuhan Pola)	4,28	0,58	Sangat Baik
3	Responsivitas dan Stabilitas Sistem (Kendali & FPS)	4,40	0,55	Sangat Baik
4	Keseimbangan Tingkat Kesulitan (<i>Game Balance</i>)	4,01	0,70	Baik
	Rata-rata Keseluruhan	4,26	0,61	Sangat Baik

Berdasarkan Tabel 8, implementasi sistem secara keseluruhan memperoleh nilai rata-rata gabungan sebesar 4,26 (Sangat Baik). Capaian ini mengonfirmasi bahwa arsitektur WFC *real-time* yang diusulkan tidak hanya efisien secara komputasional, tetapi juga berhasil memberikan pengalaman bermain yang berkualitas tinggi.

Secara lebih rinci, indikator Responsivitas dan Stabilitas Sistem memperoleh skor tertinggi (4,40 dengan SD 0,55). Hal ini membuktikan keandalan metode *sliding window grid* asinkron (menggunakan *C# Coroutines*) dan *Dynamic Object Pooling* dalam menekan latensi. Responden tidak merasakan adanya interupsi *stuttering* atau penurunan laju bingkai (FPS) pada saat *engine* merakit *chunk* tingkat yang baru, sehingga permainan terasa sangat responsif dan mulus.

Pada aspek Estetika Visual dan Kohesi Spasial (4,35), responden memberikan penilaian yang sangat positif terhadap keluwesan transisi lingkungan. Implementasi matriks *Socket Registry* sukses menerjemahkan batasan ruang menjadi koridor yang logis, sehingga fenomena *mesh clipping* (objek tumpang tindih) yang kerap merusak imersi pada PCG berbasis pengacakan

(*randomization*) dapat tereliminasi. Hal ini berbanding lurus dengan indikator Tingkat *Replayability* (4,28), di mana keragaman formasi rintangan hasil fase propagasi WFC dinilai sangat adaptif dalam memitigasi *pattern fatigue* (kejuhan pemain terhadap pola yang berulang).

Meskipun demikian, indikator Keseimbangan Tingkat Kesulitan mencatat skor terendah, yakni 4,01 (Baik) dengan nilai standar deviasi tertinggi (0,70). Nilai standar deviasi yang cukup lebar ini mengindikasikan adanya variasi persepsi yang signifikan di antara partisipan. Melalui wawancara pasca-pengujian, diketahui bahwa algoritma WFC—yang murni beroperasi berdasarkan entropi ruang—terkadang membangkitkan kluster rintangan dengan kepadatan tinggi (lonjakan kesulitan/*difficulty spike*) secara tiba-tiba. Karena sistem generasi ini belum dilengkapi dengan modul *Dynamic Difficulty Adjustment* (DDA) yang mampu mendeteksi tingkat keahlian (*skill level*) pemain secara otonom, sebagian responden yang belum terbiasa merasa kurva kesulitannya sesekali terasa tidak seimbang.

Secara keseluruhan, analisis data empiris ini menegaskan bahwa otomatisasi tata letak aset lingkungan menggunakan algoritma WFC sangat berhasil mengkonstruksi dunia tanpa akhir (*endless*) yang koheren, variatif, dan stabil secara teknis. Temuan terkait fluktuasi tingkat kesulitan ini sekaligus memberikan landasan empiris yang kuat untuk mengarahkan tahapan peta jalan (*road map*) penelitian tahun berikutnya, yakni pengintegrasian WFC dengan model *Machine Learning* adaptif.

4. Kesimpulan

Sintesis dari hasil implementasi dan pengujian mengonfirmasi bahwa penerapan *Wave Function Collapse* sebagai inti arsitektur *Procedural Content Generation* (PCG) secara *real-time* adalah sebuah terobosan rekayasa yang sangat layak. Secara teoretis, penelitian ini berhasil membuktikan bahwa fenomena keruntuhan fungsi gelombang (superposisi ke probabilitas mutlak) dapat dijinakkan di dalam lingkungan komputasi yang sangat sensitif terhadap waktu (*time-critical environment*) seperti *game endless runner*.

Kunci keberhasilannya terletak pada pemisahan antara penyelesaian *constraints* spasial (logika) dan instansiasi *mesh* geometri (grafis). Namun, penelitian ini juga menyingkap limitasi inheren dari WFC konvensional. Algoritma ini bersifat *blind-local*; artinya ia memecahkan masalah probabilitas ketetanggaan demi ruang yang valid, tanpa memiliki visi hierarkis terhadap kurva kesulitan permainan secara global. Inilah yang menyebabkan anomali pada respons pemain terhadap lonjakan kesulitan mendadak (*difficulty spike*).

Berdasarkan hasil perancangan arsitektur, implementasi sistem, dan pengujian empiris yang telah diuraikan pada bab-bab sebelumnya, penelitian ini merumuskan beberapa kesimpulan utama sebagai berikut:

1. Efektivitas WFC dalam Lingkungan *Real-Time* Asinkron,

Penelitian ini berhasil membuktikan bahwa algoritma *Wave Function Collapse* (WFC), yang secara konvensional digunakan untuk pra-komputasi tata letak statis, dapat direkayasa ulang untuk beroperasi secara *real-time* dalam *game endless runner* 3D. Penggunaan pendekatan *sliding window grid* yang dikombinasikan dengan eksekusi asinkron berbasis *C# Coroutines* terbukti krusial dalam mendistribusikan beban komputasi entropi. Sistem mampu merakit area secara *on-the-fly* di latar belakang tanpa memicu interupsi visual (*stuttering*) pada *main thread* permainan.

2. Stabilitas Kinerja Komputasi dan Manajemen Memori,

Optimalisasi *Procedural Content Generation* (PCG) tidak terbatas yang diusulkan berhasil mencapai efisiensi waktu eksekusi dengan rata-rata 35,1 ms per generasi *chunk* tingkat. Modul *Dynamic Object Pooling Manager* yang dirancang secara terintegrasi mampu memitigasi lonjakan alokasi memori dan beban *Garbage Collection* yang berlebihan. Hal ini menjamin stabilitas perangkat lunak secara konsisten, mempertahankan laju bingkai (*frame rate*) di angka 60 FPS pada skenario beban komputasi rendah hingga menengah.

3. Integritas Validasi Spasial dan Navigabilitas Lintasan

Penerapan *Socket Registry* sebagai regulator aturan ketetanggaan (*adjacency constraints*) menghasilkan akurasi transisi spasial yang sangat tinggi (99,43%). Pemangkasan probabilitas deterministik pada fase propagasi algoritma sukses menjaga rasio kesalahan geometris seperti

mesh clipping dan rongga kosong di level yang sangat minimal (0,56%). Secara fungsional, sistem WFC ini memberikan jaminan keamanan navigasi sebesar 99,73%, memastikan entitas karakter selalu mendapati lintasan tanpa akhir yang logis, aman, dan bebas dari jalur buntu (*dead-end*) absolut.

4. Implikasi terhadap Pengalaman Pengguna (*Player Experience*)

Secara kualitatif dan psikologis, otomatisasi prosedural yang dihasilkan oleh WFC terbukti mampu mengeliminasi keletihan pola (*pattern fatigue*) yang sering menjangkiti *game endless runner* linier. Variasi estetika yang dinamis berkorelasi positif dengan tingginya nilai kepuasan dan *replayability* pemain (skor rata-rata 4,26 dari skala 5,00). Meskipun demikian, temuan evaluasi juga mengungkap adanya lonjakan kurva kesulitan (*difficulty spike*) yang diakibatkan oleh keterbatasan WFC murni dalam memprediksi kelelahan motorik pemain pada area berdensitas tinggi.

Sebagai saran untuk penelitian selanjutnya, pengembangan sistem dapat dilakukan dengan:

1. Integrasi dengan *Machine Learning*: Disarankan untuk mengawinkan arsitektur deterministik WFC ini dengan algoritma *Reinforcement Learning* (RL) atau sistem penyesuaian kesulitan dinamis (*Dynamic Difficulty Adjustment*). Lapisan kecerdasan buatan tambahan ini akan membantu sistem secara otomatis mengubah bobot probabilitas aset secara aktual berdasarkan profil kemahiran dan metrik performa masing-masing pemain untuk menjaga titik keseimbangan *flow*.
2. Ekspansi Topologi Kompleks: Penelitian selanjutnya dapat mengeksplorasi modifikasi mesin WFC agar tidak hanya memproses tata letak pada grid bidang datar (2D *plane* dengan *mesh* 3D), tetapi juga mencakup vertikalitas tingkat lanjut, seperti jalur bersusun (*multi-layered paths*) atau transisi pada geometri non-Euclidean dalam ruang virtual berskala lebih masif.

Daftar Pustaka

- [1] F. Yiu et al., "A Markovian Framing of WaveFunctionCollapse for Procedurally Generating Aesthetically Complex Environments," arXiv preprint arXiv:2509.09919, 2025.
- [2] L. Marlina and Aswandi, "Android-Based Endless Runner Game Learning Media Innovations To Improve Students' Mathematical Literacy," Jurnal Eduscience, vol. 13, no. 1, pp. 128–138, 2026, doi: 10.36987/jes.v13i1.7513.
- [3] H. Kim et al., "Automatic Generation of Game Content using a Graph-based Wave Function Collapse Algorithm," in Proc. IEEE Conf. Games (CoG), London, U.K., pp. 1–4, 2019, doi: 10.1109/CIG.2019.8848019.
- [4] K. Hong, "Developing A 2d Platform Endless Runner Game With Unity Engine," M.T. thesis, Vaasa University of Applied Sciences, Finlandia, 2023.
- [5] A. C. Jovan et al., "Development of Endless Runner Game UNTARUN on Web Platform," International Journal of Application on Sciences, Technology and Engineering (IJASTE), vol. 1, no. 2, pp. 748–754, , 2023, doi: 10.24912/ijaste.v1.i2.748-754.
- [6] M. S. Yazir, E. T. Tosida, and L. Karlitasari, "Endless Run Based Medicinal Plant Educational Game Development," International Journal of Global Operations Research, vol. 3, no. 2, pp. 64–73, 2022, doi: 10.47194/ijgor.v3i2.138.
- [7] I. Reinholdsson, "Enhancing the Wave Function Collapse Algorithm with Reinforcement Learning for Controllable Procedural Content Generation," Master's thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2024.
- [8] Y. Nie et al., "Extend Wave Function Collapse to Large-Scale Content Generation," in Proc. IEEE Conf. Games (CoG), Boston, MA, USA, pp. 1–8, 2023, doi: 10.1109/CoG57401.2023.10333214.
- [9] T. Ngo, "Implementation of the Wave Function Collapse Algorithm in SAT with Custom Propagator," Master's thesis, Aalto University of Science, Espoo, Finlandia, 2025.

- [10] M. Babin and M. Katchabaw, "Leveraging Reinforcement Learning and WaveFunctionCollapse for Improved Procedural Level Generation," in Proc. 16th Int. Conf. Found. Digital Games (FDG), 2021, doi: 10.1145/3472538.347254.
- [11] Q. E. Morris, "Modifying Wave Function Collapse for more Complex Use in Game Generation and Design," Master's thesis, Trinity University, San Antonio, Texas, 2021.
- [12] G. G. Manners, "Procedural Content Generation in 3 Dimensions using Wave Function Collapse in Minecraft," Master's thesis, Trinity University, San Antonio, Texas, 2023.
- [13] V. Cerny, "Procedural Generation of Endless Runner Type of Video Games," Master's thesis, Univerzita Karlova, Czech Republic, 2018.
- [14] Z. Zhang et al., "Towards Wave Function Collapse using Optimization with Quantum Algorithms," SIGGRAPH Asia 2024 Technical Communications, pp. 1–4, 2024.
- [15] Y. Li, "Wave Function Collapse and Spontaneous Symmetry Breaking," Preprints.org, 2024, doi: 10.20944/preprints202409.2405.v4.
- [16] I. Karth and A. M. Smith, "WaveFunctionCollapse Content Generation via Constraint Solving and Machine Learning," IEEE Trans. Games, vol. 14, no. 14, pp. 364–376, 2021, doi: 10.1109/TG.2021.3076368.
- [17] D. Cheng, H. Han, and G. Fei, "Automatic Generation of Game Levels Based on Controllable Wave Function Collapse Algorithm," (eds) Entertainment Computing – ICEC 2020. ICEC 2020. Lecture Notes in Computer Science(), vol 12523. Springer, Cham, 2020, doi: 10.1007/978-3-030-65736-9_3.
- [18] R. Bailly and G. Levieux, "Genetic-WFC: Extending Wave Function Collapse With Genetic Search," in IEEE Transactions on Games, vol. 15, no. 1, pp. 36–45, March 2023, doi: 10.1109/TG.2022.3192930.
- [19] Y. Nie, S. Zheng, Z. Zhuang and J. Togelius, "Nested Wave Function Collapse Enables Large-Scale Content Generation," in IEEE Transactions on Games, vol. 16, no. 4, pp. 892–902, Dec. 2024, doi: 10.1109/TG.2024.3377637.