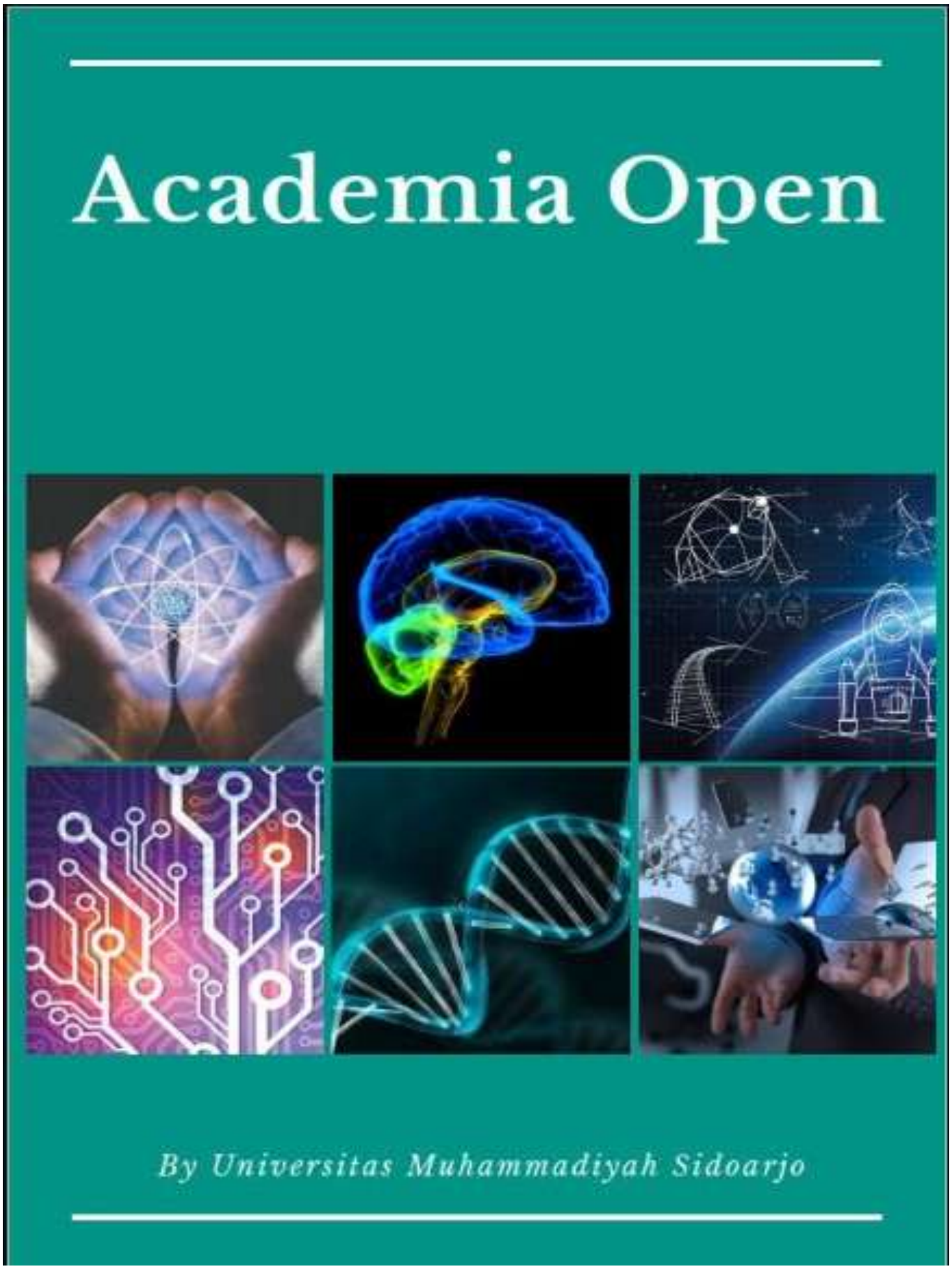
Academia Open

Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129



ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open

Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

Table Of Contents

Academia Open

Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

Originality Statement

The author[s] declare that this article is their own work and to the best of their knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the published of any other published materials, except where due acknowledgement is made in the article. Any contribution made to the research by others, with whom author[s] have work, is explicitly acknowledged in the article.

Conflict of Interest Statement

The author[s] declare that this article was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright Statement

Academia Open

Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

EDITORIAL TEAM

Academia Open

Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

Article information

Check this article update (crossmark)

Check this article impact (*)

Save this article to Mendeley

(*) Time for indexing process is various, depends on indexing database platform

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open

Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

Robustness Evaluation of Gaming Performance Against Input Rate Variations in Procedurally Generated Roguelike on Godot Engine: Evaluasi Robustness Performa Game Roguelike terhadap Variasi Input Rate Menggunakan Monkey Testing pada Godot Engine

Rizky Aji Santoso, rizky.aji.2205356@students.um.ac.id (*)
Program Studi Teknik Informatika, Universitas Negeri Malang, Indonesia

Harits Ar Rosyid, harits.ar.ft@um.ac.id
Program Studi Teknik Informatika, Universitas Negeri Malang, Indonesia

(*) Corresponding author

Abstract

General Background: System resilience under high interaction frequency is essential for maintaining real-time game stability. **Specific Background:** This study evaluates the Dodge and Deflect roguelike game on Godot Engine 4.4.1 using Monkey Testing with controlled input rate variation in a low-end virtual environment. **Knowledge Gap:** Quantitative analysis of input intensity on engine stability remains limited compared to playability-focused studies. **Aims:** This study aims to identify performance degradation patterns and critical stability thresholds driven by input rate escalation. **Results:** A strong negative correlation (r = -0.93) is found between input intensity and system stability, with performance declining from Level 11 to Level 4 under extreme conditions, accompanied by FPS drops and increased freeze events, while memory usage remains stable. **Novelty:** The study applies controlled Monkey Testing to isolate input rate as the main stress factor in a procedurally generated roguelike setting. **Implications:** The findings provide an empirical basis for optimizing event handling and defining minimum system requirements.

Highlights
- Inverse pattern between interaction density and system endurance
- Failure threshold identified through reduced progression capability
- Processing constraints emerge as dominant limitation under stress

Keywords
Monkey Testing; Godot Engine; Input Rate; Performance Degradation; Game Robustness

Published date: 2026-05-01

PENDAHULUAN

Penjaminan kualitas atau *Quality Assurance* (QA) pada industri game modern menghadapi tantangan besar dalam mendeteksi anomali perilaku sistem di bawah interaksi pengguna yang masif. Ketergantungan pada pengujian manual dinilai sangat tidak efisien karena keterbatasan fisik manusia dalam melakukan interaksi frekuensi tinggi secara terus-menerus, yang sering kali menyebabkan degradasi performa luput dari pengamatan sebelum produk dirilis. Ketidakkonsistenan manusia dalam melakukan pengujian regresi dapat menyebabkan ditemukannya *bug* kritis, kebocoran memori, atau degradasi performa yang merugikan secara finansial serta merusak reputasi produk. Walaupun otomatisasi pengujian menggunakan agen cerdas berbasis *Reinforcement Learning* (RL) mulai berkembang, mayoritas penelitian masih berfokus pada kemampuan agen dalam menyelesaikan permainan (*playability*), sementara analisis kuantitatif mengenai pengaruh intensitas input terhadap stabilitas mesin game (*game engine*) masih sangat jarang dilakukan. Fenomena degradasi performa akibat lonjakan kejadian (*event*) dalam skenario interaksi *real-time* yang ekstrem merupakan parameter krusial yang perlu diukur untuk menjamin kelayakan sebuah game di berbagai kelas perangkat.

Pengujian performa menjadi semakin kompleks pada game yang mengadopsi teknologi *Procedural Content Generation* (PCG), terutama pada genre Roguelike, karena setiap sesi menawarkan variasi tantangan yang luas melalui produksi level dan tata letak secara otomatis. Kompleksitas yang muncul dari algoritma prosedural ini sering kali menciptakan ruang keadaan (*state space*) yang sangat luas dan tidak terprediksi, yang memicu tantangan besar bagi proses QA dalam mendeteksi anomali perilaku sistem. Penelitian ini tidak bertujuan mengevaluasi algoritma PCG tersebut, melainkan memanfaatkan karakteristik Roguelike tersebut sebagai medan uji dinamis untuk mengukur ambang batas ketahanan mesin game terhadap beban interaksi pengguna. Dengan mengatur variasi frekuensi input dari rendah hingga sangat tinggi serta menjaga urutan level tetap sebagai variable control, penelitian ini bertujuan memetakan profil game

Secara empiris pada perangkat kelas bawah. Secara khusus, penelitian ini mengevaluasi ketahanan performa game roguelike Dodge and Defletc berbasia Godot Engine 4.4.1 melalui manipulasi inpu rate. Focus utamanya adalah merancang dan menerapkan kerangka Monkey Testing otomatis untuk melalukan stress testing pada engine secara efisien tanpa bergantung pada skrip perilaku yang kompleks. Dengan mengontrol frekuensi input, penelitian ini memetakan profil performa game dalam menangani tumpukan kejadian (*event queue*) dan logika fisika proyektil yang intens guna memberikan pemahaman mendalam mengenai titik kritis di mana sebuah mesin game mulai kehilangan stabilitas fungsionalnya akibat beban interaksi yang berlebihan.

Selain memberikan kontribusi metodologis, penelitian ini dirancang secara khusus untuk mensimulasikan skenario terburuk (*worst-case scenario*) dengan menjalankan seluruh rangkaian pengujian di dalam mesin virtual Oracle Virtual Box yang memiliki sumber daya terbatas. Pendekatan ini diambil sebagai bentuk validasi performa untuk memberikan jaminan stabilitas bagi segmen pengguna yang menggunakan perangkat keras kelas bawah (*low-end device*) dengan memori sebesar 4GB. Dengan mensimulasikan beban kerja tinggi pada lingkungan yang terbatas, hasil penelitian ini diharapkan dapat menjadi rujukan praktis bagi pengembang game independen dalam melakukan optimasi performa serta menentukan standar spesifikasi minimum game secara objektif berdasarkan data empiris.

Tinjauan Pustaka

Metodologi pengujian game secara otomatis telah mengalami pergeseran paradigma dari pengujian berbasis skrip statis menuju pengujian berbasis perilaku dinamis yang lebih adaptif terhadap perubahan konten. Politowski dkk. [2] dalam survei luasnya menegaskan bahwa industri game saat ini masih berada pada tahap transisi di mana otomatisasi menjadi kebutuhan mendesak untuk menekan biaya produksi yang semakin membengkak akibat kompleksitas perangkat lunak. Tantangan utama yang diidentifikasi adalah rendahnya tingkat adopsi alat pengujian otomatis di kalangan praktisi dibandingkan dengan industri perangkat lunak konvensional, yang menyebabkan proses penjaminan kualitas game sering kali masih bersifat *ad-hoc* dan sangat bergantung pada intuisi penguji manusia [15]. Salah satu pendekatan yang mulai banyak diadopsi untuk menjembatani celah ini adalah *Behaviour-Driven Development* (BDD), yang memungkinkan pengembang untuk mendefinisikan skenario pengujian berdasarkan spesifikasi perilaku pengguna, meskipun implementasinya pada mesin game masih memerlukan penyesuaian teknis yang sangat kompleks guna menangani sifat non-deterministik game [11].

Dalam konteks game yang memanfaatkan mekanisme *Procedural Content Generation* (PCG), proses pengujian menjadi jauh lebih menantang karena setiap sesi permainan menghasilkan tata letak, musuh, dan tantangan yang berbeda-beda secara stokastik [7]. Kalafatis dkk. [1] mengusulkan penggunaan agen cerdas berbasis *Deep Reinforcement Learning* (DRL) sebagai kerangka kerja modular untuk mengevaluasi kualitas konten yang dihasilkan oleh algoritma PCG secara otomatis. Meskipun agen cerdas mampu memberikan evaluasi yang mendalam, Gisslén dkk. [4] mencatat bahwa melatih agen tersebut dalam lingkungan yang dinamis membutuhkan sumber daya komputasi dan waktu yang sangat besar, sehingga sering kali tidak terjangkau bagi pengembang game independen atau tim dengan anggaran terbatas. Oleh karena itu, diperlukan metode alternatif yang lebih ringan namun tetap memiliki daya jelajah yang luas dalam mengeksplorasi ruang keadaan game yang tidak terbatas, seperti penggunaan metode *Monkey Testing*.

Monkey Testing muncul sebagai solusi pengujian fungsional yang mengandalkan input acak secara terus-menerus untuk memicu perilaku sistem yang tidak terduga atau tidak terdefinisi dalam skenario penggunaan normal. Dawson [13] menyatakan bahwa metode ini sangat efektif dalam menemukan *edge cases* atau kesalahan logika fatal yang sering kali terabaikan oleh penguji manusia yang cenderung memiliki pola bermain yang terstruktur dan terprediksi. Keunggulan utama dari *Monkey Testing* terletak pada kemampuannya untuk melakukan ribuan interaksi dalam waktu singkat tanpa memerlukan model perilaku yang rumit, yang secara teknis dapat membebani antrian kejadian (*event queue*) pada sistem

Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

untuk menguji ambang batas stabilitasnya [19]. Konsep pengujian perilaku acak ini bahkan telah diadaptasi ke dalam domain sistem basis data melalui *Monkey DB* untuk memvalidasi kebenaran data di bawah tingkat isolasi yang lemah, yang menunjukkan efektivitas metode "acak" dalam mendeteksi anomali pada sistem perangkat lunak yang kompleks [19].

Pemilihan Godot Engine sebagai subjek pengujian dalam penelitian ini didasarkan pada karakteristik teknisnya sebagai mesin game sumber terbuka yang memiliki profil performa yang unik namun menantang. Öztürk [5] dalam studinya mengenai faktor *maintainability* pada mesin game sumber terbuka mencatat bahwa Godot seringkali diklasifikasikan sebagai sistem yang memerlukan perhatian ekstra pada optimasi kode karena karakteristik manajemen skripnya. Penggunaan bahasa skrip internal seperti GDScript memungkinkan integrasi sistem pengujian yang menyatu secara organik dengan logika game, namun stabilitas visualnya sangat bergantung pada konsistensi *frame timing*. Klein dkk. [12] menekankan bahwa variasi pada *frame timing*—yang dipicu oleh lonjakan input atau beban kalkulasi fisika proyektil—dapat menyebabkan degradasi pengalaman pengguna secara drastis, sehingga evaluasi terhadap ketahanan sistem di bawah tekanan input frekuensi tinggi menjadi parameter kualitas yang sangat vital dalam pengembangan game modern.

Eksperimen performa pada sistem game juga harus mempertimbangkan kondisi batasan perangkat keras melalui skenario pengujian stres yang terkontrol. Rezende dkk. [17] menunjukkan bahwa pengujian beban pada CPU melalui algoritma intensif mampu mengidentifikasi titik hambat (*bottleneck*) sistem secara akurat, terutama ketika aplikasi dijalankan dalam lingkungan dengan sumber daya terbatas. Selain itu, penggunaan agen otomatis dalam pengujian stres terbukti dapat membantu pengembang dalam menangkap masalah penyeimbangan aturan game serta strategi dominan yang mungkin merusak pengalaman bermain [18]. Dengan menggabungkan teori pengujian interaksi *real-time* dan benchmark performa pada tingkat mesin, penelitian ini memposisikan manipulasi *input rate* sebagai variabel penekan utama untuk melihat sejauh mana mesin game Godot dapat mentoleransi lonjakan interaksi tanpa mengalami kegagalan

fungsional. Sintesis dari berbagai literatur ini menunjukkan bahwa meskipun otomatisasi game masih memiliki perjalanan panjang [15], pendekatan praktis seperti *Monkey Testing* dengan parameter frekuensi input terkontrol menawarkan solusi yang efisien untuk menilai *robustness* game Roguelike di masa depan.

METODOLOGI PENELITIAN

Penelitian ini menerapkan metode eksperimental kuantitatif yang berfokus pada pengujian stres (*stress testing*) fungsional untuk mengukur ketangguhan (*robustness*) sistem. Arsitektur pengujian dirancang untuk beroperasi sepenuhnya di dalam lingkungan virtual yang terkontrol guna mensimulasikan kondisi perangkat keras kelas bawah (*low-end device*). Penggunaan Oracle Virtual Box sebagai lapisan virtualisasi memungkinkan peneliti untuk membatasi sumber daya sistem secara presisi, yaitu dengan mengalokasikan memori RAM sebesar 4GB dan kecepatan pemrosesan CPU sebesar 2GHz. Pendekatan *worst-case environment* ini dipilih untuk memastikan bahwa setiap temuan mengenai stabilitas performa dapat dianggap sebagai standar minimum ketahanan game "Dodge and Deflect" ketika dihadapkan pada skenario penggunaan yang paling menantang [17].

Pemilihan satu subjek game (*single-case study*) dilakukan untuk menjaga konsistensi variabel arsitektur perangkat lunak, sehingga fluktuasi performa yang teramati murni disebabkan oleh manipulasi *input rate*. Fokus pada satu model game memungkinkan analisis longitudinal yang mendalam terhadap mekanisme penanganan kejadian (*event handling*) pada Godot Engine tanpa adanya bias dari perbedaan logika pemrograman antar game. Pendekatan ini merupakan standar dalam evaluasi teknis mesin game untuk mendapatkan data *benchmark* yang terkontrol.

Gambar 1. *Flowchart Metodologi Pengujian Robustness*

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Academia Open
Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

Lingkungan pengujian dalam penelitian ini disusun menggunakan mekanisme *Procedural Content Generation* (PCG) dengan pendekatan *randomized level sequencing* yang terdiri dari delapan varian modul level statis. Meskipun sistem memiliki kemampuan untuk menghasilkan urutan level yang acak secara tak terbatas, penelitian ini menerapkan prosedur isolasi variabel yang ketat guna memastikan validitas data empiris. Setiap skenario pengujian, mulai dari tingkat *Low* hingga *Extreme*, dijalankan menggunakan *seed* yang sama sehingga bot menghadapi urutan modul level yang identik di setiap sesi tes. Standarisasi urutan ini sangat krusial agar fluktuasi performa yang terekam, seperti penurunan FPS atau peningkatan penggunaan memori, murni disebabkan oleh manipulasi intensitas *input rate* dan bukan dipicu oleh perbedaan kompleksitas struktur antar level atau perbedaan urutan kemunculan *map* yang dihasilkan secara prosedural.

Pusat dari metodologi ini adalah pengembangan framework pengujian otomatis berbasis Monkey Testing yang diintegrasikan langsung ke dalam mesin Godot menggunakan bahasa skrip GDScript. Berbeda dengan pengujian berbasis AI-agents yang memerlukan pelatihan model yang kompleks [1], [4], Monkey Testing yang dikembangkan dalam penelitian ini bekerja dengan mensimulasikan kejadian input keyboard dan mouse secara stokastik. Bot ini dirancang untuk memicu berbagai fungsi permainan, mulai dari pergerakan karakter hingga interaksi proyektil, guna memastikan bahwa antrian kejadian (event queue) pada mesin game terisi secara konstan. Implementasi ini memungkinkan cakupan pengujian yang luas terhadap logika collision detection dan manajemen objek di dalam memori tanpa memerlukan intervensi manual dari penguji manusia [13].

Variabel utama yang dimanipulasi dalam penelitian ini adalah input rate atau frekuensi kejadian input yang dikategorikan ke dalam empat tingkat stress: Low, Medium, High, dan Extreme. Setiap tingkat didefinisikan oleh interval waktu milidetik antar input, di mana tingkat Low mensimulasikan interaksi manusia normal dengan jeda yang cukup, sementara tingkat Extreme dirancang untuk membombardir mesin game dengan ribuan perintah dalam waktu singkat Pembagian ini bertujuan mengidentifikasi titik jenuh saat engine Godot mulai mengalami penurunan performa visual atau gangguan sinkronisasi frame. Dengan Pengendalian intensitas secara berkelanjutan, hubungan antara kepadatan input dan stabilitas system dapat dianalisis secara empiris.

Data dikumpulkan melalui system logger internal yang mencatat metrik secara realtime setiap 50 input, untuk menjaga keseimbangan antara detail data dan bebasn system. Parameter yang dikukur meliputi rata-rata FPS, FPS minimun (stuttering), penggunaan memori (MB), dan jumla freeze. Data disimpan dalam struktur dinamis lalu diekspor otomatis ke CSV, dengan timestamp presisi untuk mengaitkan perubahan performa dengan level yang dihasilkan.

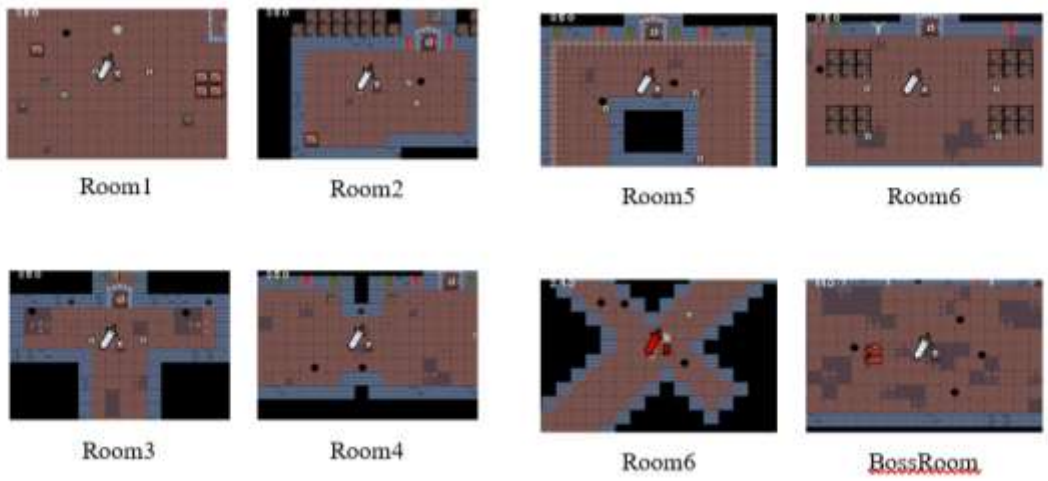
Validasi dilakukan melalui perbandingan profil performa antar tingkat stress menggunakan statistic desktiptif, dengan memastikan setiap pengujian berjalan hingga mencapai batas kegagalan atau menyelesaikan siklus permainan. Lingkungan virtual dibersihkan kembali ke keadaan awal (fresh state) setiap kali berpindah ke tingkat stress yang berbeda untuk menghindari bias data akibat fragmentasi memori atau residu proses sebelumnya. Metodologi yang ketat ini dirancang untuk menghasilkan data yang memiliki tingkat reprodusibilitas tinggi, sehingga pengembang lain dapat mereplikasi skema

Academia Open
Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

pengujian stres ini pada proyek game berbasis Godot lainnya guna menjamin kualitas sebelum tahap publikasi [15], [20].

HASIL DAN PEMBAHASAN

Gambar 2. *Representasi Delapan Varian Modul Level sebagai Variabel Kontrol Lingkungan.*



A. Evaluasi Metrik Performa Kuantitatif

Evaluasi metrik performa kuantitatif dilakukan secara sistematis untuk memetakan perilaku mesin game Godot ketika dihadapkan pada variasi beban interaksi. Guna memberikan gambaran mengenai karakteristik lingkungan yang menjadi medan uji, Gambar X (Silakan masukkan gambar kolase 8 varian modul level Anda) menyajikan representasi visual dari kedelapan varian modul level yang menyusun dunia permainan dalam "Dodge and Deflect". Setiap modul dirancang manual dengan variasi rintangan dan kepadatam proyektil untuk merepesentasikan kompleksitas roguelike. Urutan level dikunci menggunakan seed yang sama di semua scenario beban lingkungan tetap identic, sehingga perubahan performa murni disebabkan oleh variasi input rate.

Data performa dikumpulkan secara sistematis untuk mengamati respons Godot 4.4.1 terhadap beban interaksi dari Monkey Testing. Hasil awal menunjukkan tiap tingkat stress menghasilkan tekanan berbeda, di mana pada intensitas rendah system masih mampu mengelola antrian event dengan baik dan bertahan lebih lama dibanding scenario. Hal ini mengkonfirmasi teori bahwa durasi operasional sebuah game sangat dipengaruhi oleh kepadatan interaksi *real-time* yang harus diproses oleh mesin pada setiap siklus *frame*-nya [12], [15]. Berdasarkan serangkaian pengujian stres yang telah dilakukan pada empat tingkat intensitas input yang berbeda, didapatkan ringkasan data performa yang mencakup aspek stabilitas visual, ketahanan level, serta efisiensi manajemen memori. Data rata-rata dari seluruh sesi pengujian dirangkum dalam Tabel 1 berikut:

Tabel 1. *Perbandingan Metrik Performa Game pada Berbagai Tingkat Stress*

Stress Level	Avg FPS	Min FPS	Max Memory (MB)	Max Freeze Count	Max Level
LOW (0)	12.75	1	93.51	176	11
MEDIUM (1)	12.08	3	91.96	46	7
HIGH (2)	24.09	7	88.9	0	4
EXTREME (3)	12.49	4	88.2	23	4

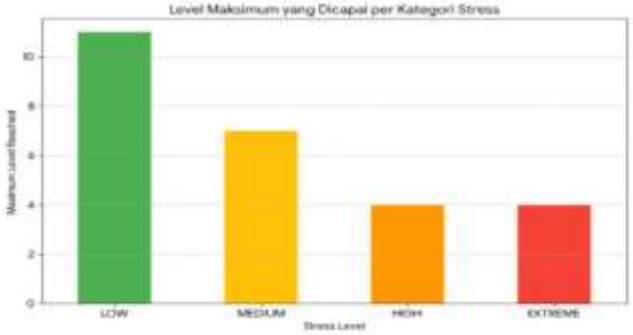
Analisis statistik menggunakan koefisien korelasi Pearson menunjukkan adanya korelasi negatif yang sangat kuat antara peningkatan *stress level* (intensitas input) dengan daya tahan sistem (level maksimum yang dicapai) dengan nilai $\mathbf{r = -0.93}$ serta nilai $\boldsymbol{p < 0.05}$, yang menunjukkan adanya signifikansi statistik. Hasil ini menunjukkan bahwa peningkatan frekuensi interaksi secara signifikan mempercepat tercapainya titik jenuh system, sehingga sesi pengujian berakhir lebih cepat akibat lonjakan beban kejadian.

Data kuantitatif pada table 1 memperlihatkan profil ketahanan system di tiap scenario, dengan pola jelas bahwa semakin tinggi intensitas input, semakin menurun kapasitan fungsional game, terutama pada level maksimun yang mampu diselesaikan. Meskipun tabel memberikan gambaran umum mengenai rata-rata performa, terdapat fluktuasi pada metrik stabilitas visual dan frekuensi kejadian layar membeku (*freeze*) yang memerlukan analisis lebih mendalam. Fenomena degradasi performa ini mengarahkan pada pentingnya melihat distribusi data secara visual guna mengidentifikasi ambang

Academia Open
Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

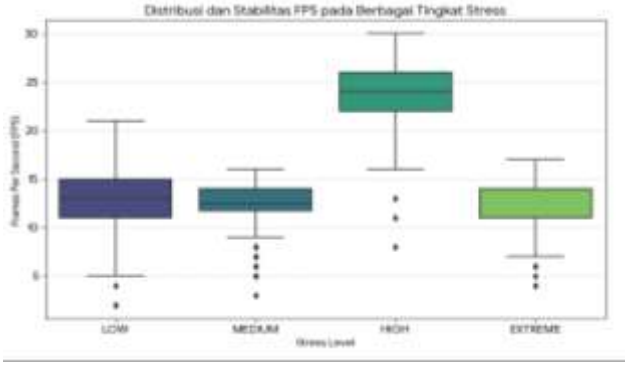
batas kegagalan sistem. Oleh karena itu, bagian selanjutnya akan membedah tren tersebut melalui serangkaian grafik untuk melihat stabilitas *frame per second* dan karakteristik *bottleneck* selama sesi pengujian berlangsung.

Gambar 3. *Level Maksimum yang Dicapai (Bar Chart)*



Grafik di atas menyajikan perbandingan tingkat keberhasilan bot dalam menyelesaikan level permainan pada setiap skenario intensitas input. Terlihat adanya degradasi fungsional yang signifikan di mana pada tingkat *Low*, bot mampu mencapai Level 11 sebelum sesi berakhir. Namun, seiring dengan peningkatan intensitas menjadi *Extreme*, daya tahan sistem menurun drastis hingga hanya mampu mencapai Level 4. Grafik di atas menyajikan perbandingan tingkat keberhasilan bot dalam menyelesaikan level permainan pada setiap skenario intensitas input. Terlihat adanya degradasi fungsional yang signifikan di mana pada tingkat *Low*, bot mampu mencapai Level 11 sebelum sesi berakhir. Namun, seiring dengan peningkatan intensitas menjadi *Extreme*, daya tahan sistem menurun drastis hingga hanya mampu mencapai Level 4. Penurunan ini memberikan bukti empiris bahwa intensitas input yang tinggi memaksa sistem mencapai ambang batas kegagalan lebih cepat, meskipun urutan level yang dilewati bot adalah identik di setiap skenario akibat penggunaan *fixed seed*.

Gambar 4. *Distribusi dan Stabilitas FPS (Box Plot)*



Distribusi nilai FPS yang disajikan dalam bentuk boxplot di atas memberikan gambaran mendalam mengenai stabilitas visual selama pengujian berlangsung. Terdapat temuan unik pada kategori High di mana nilai rata-rata FPS justru tercatat lebih tinggi dibandingkan kategori lainnya, namun dengan sebaran (interkuartil) yang lebih lebar.

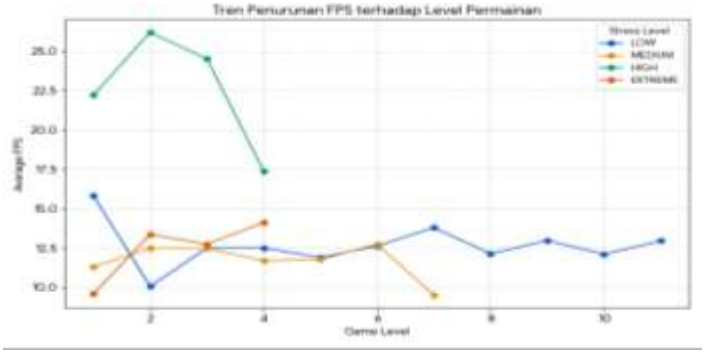
Fenomena ini menunjukkan adanya ketidakstabilan frame timing (frame jitter) yang mulai muncul sebelum sistem mencapai titik saturasi penuh. Dengan kata lain, HIGH bukan kondisi optimal, tetapi kondisi transisi menuju overload sistem. Sebaliknya, pada kategori Extreme, sebaran nilai FPS menjadi sangat sempit di kisaran rendah, yang menunjukkan bahwa mesin game telah mencapai titik jenuh di mana ia tidak lagi mampu melakukan kompensasi terhadap frame timing [12]. Stabilitas FPS yang rendah pada beban ekstrem ini mengindikasikan bahwa *engine* memprioritaskan penyelesaian antrian *event* daripada kelancaran *rendering* visual.

B. Analisis Degradasi Performa dan Frame Timing

Fenomena degradasi performa yang diamati dalam penelitian ini menunjukkan bahwa fluktuasi *input rate* secara langsung memperburuk inkonsistensi waktu antar *frame* (*jitter*). Dalam lingkungan virtual Oracle Virtualbox yang memiliki sumber daya CPU terbatas (2GHz), setiap lonjakan input dari *Monkey Testing* menyebabkan penundaan pada proses *update* logika fisika dan *rendering loop*. Ketidakmampuan mesin untuk mensinkronkan ribuan kejadian input proyektil dengan tampilan visual menyebabkan penurunan *Min FPS* yang drastis, yang secara langsung merusak pengalaman bermain seandainya diuji oleh manusia [12], [16]. Penurunan ini juga berkaitan dengan meningkatnya jumlah kalkulasi collision detection dan update posisi objek secara simultan dalam satu frame, yang secara eksponensial meningkatkan beban komputasi seiring bertambahnya jumlah entitas aktif di layar. Analisis ini diperkuat oleh fakta bahwa peningkatan level pada game Roguelike sering kali disertai dengan peningkatan jumlah objek aktif di layar, yang secara kumulatif memperberat beban komputasi per *frame*.

Academia Open
Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

Gambar 5. *Tren Penurunan FPS terhadap Level (Line Plot)*



Grafik tren di atas menggambarkan bagaimana performa visual mengalami penurunan yang berfluktuasi seiring dengan bertambahnya level permainan. Pada skenario *Low* dan *Medium*, grafik menunjukkan pola penurunan yang lebih landai, yang mencerminkan ketersediaan *headroom* komputasi bagi sistem untuk menangani konten prosedural [1]. Namun, pada garis tren *Extreme*, performa visual sudah berada di bawah ambang kelancaran (15 FPS) sejak level awal dan terus mengalami tekanan hingga sesi terhenti. Perpotongan garis pada grafik ini secara visual memetakan "titik kritis" stabilitas, di mana beban interaksi yang melampaui kapasitas pemrosesan CPU menyebabkan kegagalan sinkronisasi sistem secara menyeluruh.

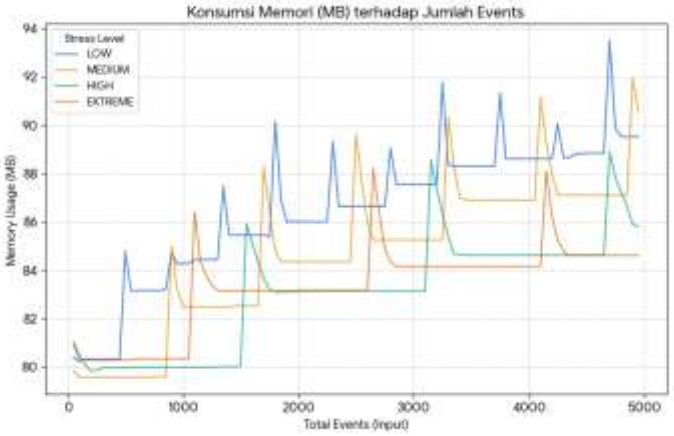
C. Manajemen Memori dan Bottleneck Sistem

Berbeda dengan aspek performa visual yang sangat fluktuatif, manajemen memori pada Godot Engine 4.4.1 dalam penelitian ini menunjukkan karakteristik yang sangat tangguh dan stabil. Penggunaan RAM di semua skenario pengujian tetap terjaga dalam kisaran 88 MB hingga 93 MB, tanpa menunjukkan adanya gejala kebocoran memori (*memory leak*) meskipun sistem dibombardir dengan ribuan input *events*. Stabilitas ini juga menunjukkan bahwa objek-objek dinamis yang dihasilkan oleh sistem PCG tidak mengalami akumulasi yang tidak terkendali di memori, sehingga dapat diasumsikan bahwa siklus hidup objek (object lifecycle) telah dikelola secara efisien oleh engine tanpa menyebabkan fragmentasi memori yang signifikan. ini membuktikan bahwa mekanisme *garbage collection* dan manajemen objek pada Godot sangat efisien dalam menangani alokasi dinamis pada game Roguelike [5]. Hal ini memberikan petunjuk krusial bahwa hambatan utama (*bottleneck*) sistem pada lingkungan *low-end* terletak pada kecepatan eksekusi logika (CPU-bound) dan bukan pada keterbatasan kapasitas penyimpanan memori (RAM-bound).

Secara implisit, hasil ini juga menguatkan teori arsitektur game real-time bahwa stabilitas memori tidak selalu berkorelasi langsung dengan penurunan performa, melainkan lebih ditentukan oleh beban komputasi CPU pada loop utama (game loop). Dengan kata lain, system tetap konsisten pada level data, namun tidak mampu menjaga kestabilan eksekusi realtime. Penurunan performa pada tingkat ekstrem menunjukkan terjadinya *input flooding* yang memicu kelebihan beban pada *event queue*. Laju input melampaui kapasitas proses *dequeue,*sehingga memperlambat *physics update loop* dan menimbulkan ketidaksinkronan antara logika fisika dan rendering. Kondisi ini diperparah oleh keterbatasan CPU dalam lingkungan virtual, yang menyebabkan *thread contention* saat menangani lonjakan input secara bersamaan.

Temuan ini menegaskan bahwa bolleneck tidak hanya berasal dari *event queue* yang overload, tetapi juga dari lemahnya sinkronisasi antar thread di lingkungan virtual, yang secara kumulatif meningkatkan latensi pemrosesan.

Gambar 6. *Konsumsi Memori (MB) terhadap Jumlah Events (Line Plot)*



Grafik konsumsi memori di atas menunjukkan pola kenaikan yang sangat teratur dan linear terhadap akumulasi jumlah kejadian input. Meskipun intensitas input dinaikkan secara ekstrem, garis memori tidak menunjukkan lonjakan eksponensial, yang memvalidasi bahwa arsitektur game "Dodge and Deflect" telah mengimplementasikan manajemen objek yang sehat. Dengan demikian, hasil ini memperkuat keterkaitan langsung dengan teori garbage collection modern yang

Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

menyatakan bahwa sistem dengan manajemen memori deterministik cenderung mempertahankan stabilitas meskipun berada pada kondisi stress tinggi.

Temuan ini sangat penting bagi pengembang game independen, karena menunjukkan bahwa strategi optimasi untuk perangkat rendah harus lebih difokuskan pada penyederhanaan kalkulasi logika fisika dan efisiensi GDScript daripada sekadar mengurangi ukuran aset atau penggunaan memori [17]. Hal ini menunjukkan bahwa lingkungan virtual berperan sebagai akselerator stress testing, karena mempercepat munculnya bottleneck dibandingkan lingkungan native.

D. Pembahasan Robustness dalam Lingkungan Virtual

Pengujian pada lingkungan mesin virtual memberikan validasi yang kuat mengenai *robustness* game di bawah kondisi operasional yang paling menantang. Dalam virtualisasi, latensi akses terhadap instruksi perangkat keras biasanya lebih tinggi dibandingkan mesin fisik, sehingga masalah stabilitas akan muncul lebih cepat dan lebih jelas [17]. Keberhasilan game ini untuk tetap berjalan hingga Level 4 pada intensitas *Extreme* tanpa mengalami *crash* total atau kegagalan memori adalah bukti ketahanan arsitektur sistem. Hal ini sejalan dengan penelitian Rezende dkk. [17] yang menyatakan bahwa *stress testing* pada lingkungan terbatas merupakan instrumen yang paling efektif untuk menjamin bahwa aplikasi dapat menangani tugas pemrosesan yang menuntut tanpa mengorbankan integritas sistem.

Secara keseluruhan, hasil pembahasan ini menegaskan bahwa penggunaan *Monkey Testing* dengan variasi frekuensi input adalah metodologi yang valid dan berbiaya rendah untuk menentukan ambang batas performa sebuah game. Data yang dihasilkan tidak hanya memberikan angka mentah, tetapi juga memberikan wawasan mendalam mengenai perilaku internal mesin game saat menghadapi beban kerja yang tidak terduga. Dengan demikian, pengujian ini dapat berfungsi sebagai sistem peringatan dini bagi pengembang untuk melakukan optimasi pada bagian-bagian game yang paling rentan terhadap lonjakan interaksi pengguna sebelum game tersebut dirilis ke publik [15], [18].

KESIMPULAN

Penelitian ini berhasil mengevaluasi *robustness* performa game Roguelike berjudul "Dodge and Deflect" yang dikembangkan menggunakan Godot Engine 4.4.1 melalui pendekatan *Monkey Testing* otomatis. Temuan utama menunjukkan bahwa intensitas interaksi pengguna (*input rate*) merupakan parameter kritis yang secara langsung berkorelasi dengan stabilitas fungsional game dalam lingkungan dengan sumber daya terbatas [19]. Seiring dengan peningkatan frekuensi input dari tingkat *Low* ke *Extreme*, sistem mengalami degradasi performa visual yang signifikan, di mana rata-rata FPS menurun drastis dan daya tahan level terhenti pada tahap awal permainan. Hal ini membuktikan bahwa pengujian stabilitas tidak boleh hanya berfokus pada penyelesaian level (*playability*), tetapi juga harus mempertimbangkan batas beban interaksi *real-time* yang mampu ditangani oleh mesin game [12], [15].

Metodologi pengujian stres (*stress testing*) dalam lingkungan virtual *worst-case* terbukti menjadi instrumen evaluasi yang efektif dan objektif. Dengan membatasi sumber daya pada Oracle Virtual Box, penelitian ini mampu mengidentifikasi ambang batas kegagalan sistem yang mungkin tidak terlihat pada perangkat keras berspesifikasi tinggi. Salah satu kontribusi penting dari penelitian ini adalah temuan mengenai karakteristik *bottleneck* pada Godot Engine, di mana penggunaan memori tetap stabil namun beban pemrosesan pada CPU menjadi hambatan utama dalam menangani antrian kejadian (*event queue*) yang masif. Fakta bahwa Godot mampu melakukan de-alokasi memori secara efisien tanpa terjadi kebocoran memori menunjukkan ketangguhan arsitektur internal *engine* tersebut dalam mengelola objek dinamis hasil PCG [5], [17].

Keberhasilan implementasi *framework* Monkey Testing berbasis GDScript ini menawarkan solusi pengujian otomatis yang berbiaya rendah dan mudah diakses bagi pengembang independen. Berbeda dengan metode AI cerdas yang memerlukan daya komputasi tinggi untuk proses pelatihan [1], [4], penggunaan input stokastik dengan frekuensi terkontrol mampu mengeksplorasi *edge cases* fungsionalitas game secara luas dalam waktu yang relatif singkat. Pendekatan ini sangat relevan dalam siklus pengembangan *agile* untuk memastikan bahwa setiap iterasi baru tidak memperkenalkan regresi performa yang dapat mengganggu pengalaman bermain pengguna pada perangkat kelas bawah [2], [20].

Meskipun memberikan hasil yang konklusif, penelitian ini memiliki keterbatasan dalam hal lingkungan pengujian yang terbatas pada virtualisasi, yang mungkin memiliki karakteristik latensi berbeda dibandingkan mesin fisik. Oleh karena itu, disarankan bagi peneliti selanjutnya untuk melakukan validasi lintas platform (*cross-platform validation*), termasuk pada perangkat *mobile* atau *web*, guna mengamati konsistensi perilaku Godot Engine di berbagai sistem operasi. Pengujian pada mesin fisik secara langsung juga akan memberikan data yang lebih akurat mengenai pengaruh *input lag* terhadap *frame timing* secara lebih mendalam [10], [12].

Untuk pengembangan lebih lanjut dari sisi teknis, optimasi pada manajemen *event handling* di Godot perlu dieksplorasi, misalnya dengan pemanfaatan *multi-threading* atau penggunaan C# untuk bagian logika yang intensif guna meminimalkan kejadian layar membeku (*freeze count*). Integrasi elemen kecerdasan buatan untuk mengarahkan perilaku *Monkey Testing* agar lebih "sadar konteks" namun tetap acak juga merupakan arah riset yang menjanjikan untuk meningkatkan cakupan pengujian (*test coverage*) pada area-area game yang memiliki kepadatan aset tinggi [18], [20]. Akhirnya, penelitian ini diharapkan dapat menjadi fondasi bagi terciptanya standar pengujian otomatis yang lebih inklusif dalam industri game, memastikan stabilitas produk dapat dijamin bagi seluruh lapisan pemain tanpa memandang spesifikasi perangkat keras yang mereka miliki.

Dengan demikian, penelitian ini menegaskan bahwa pengujian berbasis variasi input rate dapat berfungsi sebagai indikator awal yang efektif dalam mendeteksi keterbatasan performa engine sebelum sistem mengalami kegagalan kritis di lingkungan produksi.

ISSN 2714-7444 (online), https://acopen.umsida.ac.id, published by Universitas Muhammadiyah Sidoarjo

Vol. 11 No. 1 (2026): June
DOI: 10.21070/acopen.11.2026.14129

UCAPAN TERIMAKASIH

Penulis menyampaikan terima kasih kepada seluruh pihak yang telah mendukung terselesaikannya penelitian ini. Apresiasi khusus diberika kepada para pembimbing atas arahan dan masukan yang berharga, serta kepada KBK Game dan Teknologi Digital Cerdas atas dukungan fasilitas penelitian. Ucapan terima kasih juga ditujukan kepada semua pihak yang terlibat dalam pengumpulan data, pengelolaan hasil, dan pengujian system. Dukungan yang diberikan sangat membantu kelancaran penelitian ini. Semoga hasil penelitian ini bermanfaat bagi pengembangan teknologi dan menjadi rujukan bagi penelitian selanjutnya.

References

[1] E. Kalafatis dkk., "A modular framework for automated evaluation of procedural content generation in serious games with deep reinforcement learning agents," IEEE Transactions on Games, 2024.
[2] C. Politowski dkk., "A Survey of Video Game Testing," arXiv preprint arXiv:2103.06431, 2021.
[3] M. R. Rodrigues dan I. H. Farias Júnior, "Adaptive Scenario Selection in Serious Games Using Finite State Machines and Agent-Based Models," 2023.
[4] L. Gisslén dkk., "Adversarial Reinforcement Learning for Procedural Content Generation," arXiv preprint arXiv:2103.04847, 2021.
[5] S. Öztürk, "Analyzing maintainability factors in open-source game engines: implications for game developers," Multimedia Tools and Applications, 2025.
[6] Z. Zhao, "Application and Problems of AI in Game Development," Proceedings of CONF-MLA, 2024.
[7] A. Vatresia dkk., "Designing a 3D Roguelike Game with Procedural Content Generation Using the Graph Grammars Method," Jurnal Teknik Informatika (JUTIF), vol. 4, no. 6, 2023.
[8] A. Sanyal dkk., "Frequency-Based Hyperparameter Selection in Games," arXiv preprint arXiv:2601.18409, 2026.
[9] M. Babin dan M. J. Katchabaw, "Game Balance Through Procedural Content Generation," FDG '25, 2025.
[10] M. Claypool dan A. Cockburn, "Game Input with Delay - Moving Target Selection Parameters," 2023.
[11] M. Mulder dan P. van den Bos, "Introducing automated testing to video game development via Behaviour-Driven Development," Journal of Object Technology, 2025.
[12] D. Klein dkk., "The Influence of Variable Frame Timing on First-Person Gaming," arXiv preprint arXiv:2306.01691, 2023.
[13] F. Dawson, "Planting World Seeds: Procedural Content Generation, Labour and Conceptual Art in Digital Games," 2024.
[14] A. Senchenko dkk., "SUPERNOVA: Automating Test Selection and Defect Prevention in AAA Video Games," arXiv preprint arXiv:2203.05566, 2023.
[15] C. Politowski dkk., "Towards Automated Video Game Testing: Still a Long Way to Go," 2022.
[16] N. Sevcenko dkk., "Theory-based approach for assessing cognitive load during time-critical resource-managing HCI," Journal on Multimodal User Interfaces, 2023.
[17] T. M. Rezende dkk., "Benchmark review and case study of stress test for a gaming computer applied in CPU," 2024.
[18] C. G. Malosto dkk., "Moving towards automated game play-testing," SBGames 2025, 2025.
[19] R. Biswas dkk., "MonkeyDB: Effectively Testing Correctness under Weak Isolation Levels," 2023.
[20] C. Wang dkk., "Leveraging LLM Agents for Automated Video Game Testing," arXiv preprint arXiv:2509.22170, 2025.