Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id

E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari

OPEN ACCESS

https://doi.org/10.35870/jikti.v3i2.2152

Pengembangan Game 3D Menggunakan Godot Engine: Pipeline Desain, Implementasi Gameplay, dan Otomatisasi Build

Sholahudin Al'ayubi [1*], Kemas Muhammad Wahyu Hidayat [2] Syarifah Aini [3]

[1*,2,3] Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Muhammadiyah Palembang, Kota Palembang, Sumatera Selatan, Indonesia.

Corresponding Email: yubimuarajauh123@gmail.com [1*] wahyu_hidayat@um-palembang.ac.id [2] syarifah_aini@um-palembang.ac.id [3]

Histori Artikel:
Dikirim 01 Oktober 2026; *Diterima dalam bentuk revisi* 10 Oktober 2026; *Diterima* 10 November 2026; *Diterbitkan* 27 Januari 2027. Semua hak dilindungi oleh Lembaga Penelitian dan Pengabdian Masyarakat (LPPM) STMIK Indonesia Banda Aceh.

Abstrak

Pengembangan game 3D memerlukan alur kerja yang terstruktur, mulai dari perancangan aset hingga implementasi mekanik permainan dan pengujian sistem. Penelitian ini bertujuan untuk merancang dan mengimplementasikan pipeline pengembangan game 3D menggunakan Godot Engine 4.5, dengan mengintegrasikan pembuatan aset, pemrograman gameplay, dan otomatisasi proses build. Pendekatan yang digunakan dalam penelitian ini adalah Research and Development (R&D), dengan proses pengembangan yang bersifat iteratif, mencakup perancangan, implementasi, pengujian, evaluasi, dan penyempurnaan. Aset 3D dibuat menggunakan Blender dan diintegrasikan ke dalam Godot Engine melalui format glTF/GLB. Mekanik permainan dikembangkan menggunakan GDScript dengan menerapkan arsitektur berbasis komponen serta desain modular. Pengujian fungsional dilakukan dengan metode Black Box Testing, sementara otomatisasi build dirancang menggunakan GitHub Actions. Hasil pengujian menunjukkan bahwa 13 skenario pengujian fungsional, yang mencakup interaksi pemain, mekanik puzzle, kecerdasan buatan musuh, antarmuka, dan alur permainan, berhasil dijalankan sesuai dengan hasil yang diharapkan. Penelitian ini menghasilkan pipeline pengembangan game 3D yang mengintegrasikan pembuatan aset, implementasi gameplay, pengujian fungsional, dan konfigurasi otomatisasi build dalam satu alur kerja. Pengukuran kuantitatif terhadap performa permainan dan durasi build masih diperlukan untuk mengevaluasi efisiensi pipeline secara lebih mendalam.

Kata Kunci: Game 3D; Blender; Black Box Testing; Continuous Integration

Abstract

The development of 3D games requires a structured workflow, ranging from asset design to the implementation of game mechanics and system testing. This research aims to design and implement a 3D game development pipeline using Godot Engine 4.5, integrating asset creation, gameplay programming, and build process automation. The approach utilized in this research is Research and Development (R&D), characterized by an iterative development process that includes design, implementation, testing, evaluation, and refinement. 3D assets are created using Blender and integrated into Godot Engine through the glTF/GLB format. Game mechanics are developed using GDScript, applying a component-based architecture and modular design. Functional testing is conducted using the Black Box Testing method, while build automation is designed using GitHub Actions. The testing results indicate that 13 functional testing scenarios, encompassing player interactions, puzzle mechanics, enemy artificial intelligence, user interfaces, and game flow, were successfully executed in accordance with the expected outcomes. This research produces a 3D game development pipeline that integrates asset creation, gameplay implementation, functional testing, and build automation configuration within a single workflow. Quantitative measurements of game performance and build duration are still necessary to evaluate the efficiency of the pipeline in greater depth.

Keyword: Sales; 3D Game; Blender; Black Box Testing; Continuous Integration

Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

1. Pendahuluan

Perkembangan teknologi game engine memberikan peluang bagi pengembang untuk menciptakan permainan digital dengan berbagai tingkat kompleksitas (Mulachela *et al.*, 2020). Game engine menyediakan berbagai fasilitas untuk mengelola aset, mengimplementasikan mekanik permainan, dan mengintegrasikan komponen perangkat lunak. Dalam pengembangan game tiga dimensi (3D), proses ini memerlukan alur kerja yang terstruktur karena melibatkan pembuatan aset visual, pengaturan lingkungan permainan, pemrograman interaksi, serta pengujian fungsionalitas sistem. Oleh karena itu, perancangan pipeline pengembangan yang sistematis menjadi salah satu aspek penting dalam proses pengembangan game (Amir *et al.*, 2017). Godot Engine merupakan game engine sumber terbuka yang dapat digunakan untuk mengembangkan game 2D maupun 3D. Engine ini menyediakan sistem berbasis node dan mendukung penggunaan GDScript untuk mengimplementasikan logika permainan (Del Gallego *et al.*, 2026). Dalam pengembangan aset 3D, Blender dapat digunakan untuk membuat model dan material yang selanjutnya diintegrasikan ke dalam Godot Engine melalui format glTF/GLB. Integrasi ini memungkinkan proses pembuatan aset dan implementasi permainan dilakukan menggunakan perangkat lunak yang berbeda dalam satu alur kerja pengembangan (Johnson, 2023). Meskipun demikian, penerapan teknologi tersebut tetap memerlukan perancangan arsitektur sistem dan prosedur pengujian agar komponen permainan dapat berfungsi sesuai kebutuhan. Penelitian terdahulu yang dibahas dalam naskah ini menunjukkan adanya variasi fokus dalam pengembangan game menggunakan Godot Engine. Penelitian oleh Alam dan Mastan (2024) membahas pengembangan game 2D berjudul *Waddle* dan pengujian penerimaan pengguna. Penelitian lain mengenai game *Escape From Basement* menitikberatkan implementasi Finite State Machine (FSM) pada kecerdasan buatan musuh. Sementara itu, penelitian oleh Jonathan *et al.* (2023) membahas pengembangan game strategi 2D berjudul *Frontier of Hell*. Ketiga penelitian tersebut memiliki fokus yang berbeda dalam aspek pengembangan permainan. Namun, berdasarkan uraian penelitian terdahulu yang digunakan dalam artikel ini, pembahasan mengenai integrasi pipeline aset 3D, implementasi mekanik gameplay, perancangan arsitektur berbasis komponen dan modular, serta otomatisasi build belum ditunjukkan sebagai satu rangkaian pengembangan yang terpadu (Sasognko, 2025).

Berdasarkan kondisi tersebut, penelitian ini berfokus pada perancangan dan implementasi pipeline pengembangan game 3D menggunakan Godot Engine 4.5 (Martial *et al.*, 2026). Pipeline yang dikembangkan mencakup pembuatan aset 3D menggunakan Blender, integrasi aset melalui format glTF/GLB, implementasi mekanik permainan menggunakan GDScript, penerapan arsitektur berbasis komponen dan desain modular, serta konfigurasi Continuous Integration (CI) menggunakan GitHub Actions (Faqih *et al.*, 2024; Pratiwi *et al.*, 2025). Tujuan dari penelitian ini adalah merancang dan mengimplementasikan pipeline pengembangan game 3D sederhana secara end-to-end (Hilmawan & Yuniati, 2024), yang mencakup pembangunan workflow aset 3D Blender, penerapan arsitektur Component-Based Entity System dan Modular Design Pattern (Guimarães & Xexéo, 2026), serta pembangunan pipeline Continuous Integration (CI) menggunakan GitHub Actions. Tujuan penelitian ini adalah menghasilkan dan mendokumentasikan alur kerja pengembangan game 3D yang mengintegrasikan tahap perancangan aset, implementasi gameplay, pengujian fungsional, dan konfigurasi otomatisasi build. Penelitian ini diharapkan dapat memberikan gambaran teknis mengenai hubungan antartahap tersebut serta menjadi referensi bagi pengembang yang ingin memahami proses pengembangan game 3D menggunakan Godot Engine. Evaluasi dalam penelitian ini difokuskan pada implementasi pipeline dan hasil pengujian fungsional, sedangkan pengukuran kuantitatif terhadap performa permainan dan efisiensi proses build menjadi aspek yang perlu dikembangkan lebih lanjut.

Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

2. Metode Penelitian

Penelitian ini mengadopsi pendekatan *Iterative Object-Oriented Development* yang mengintegrasikan prinsip pengembangan berorientasi objek dengan siklus *Agile Development*. Metode ini dirancang melalui tahapan *Pre-Production*, *Production*, dan *Post-Production* yang dihubungkan melalui siklus iteratif *Design* → *Implement* → *Test* → *Evaluate* → *Refine* untuk memastikan setiap komponen fungsional dan pipeline terintegrasi dengan baik.(Wingqvist *et al.*, 2022)

2.1 Tahapan Pengembangan dan Hubungan Antarmetode

Alur pengembangan game dan integrasi pipeline secara terstruktur dibagi ke dalam tiga fase utama:

1. Tahap Pra-Produksi (*Pre-Production*): Tahap ini berfokus pada analisis kebutuhan, penyusunan *Game Design Document* (GDD), perancangan skenario alur permainan (*flowchart* dan *use case*), serta perancangan skenario otomatisasi *build* Continuous Integration (CI).
2. Tahap Produksi (*Production*): Tahap pembuatan aset visual 3D (pemodelan, pemetaan UV, *texturing*, dan *rigging*) pada Blender 3D. Aset kemudian diekspor ke format glTF 2.0 (.gltf/.glb) dan diimpor ke Godot Engine 4.5. Pada tahap ini, logika permainan dikembangkan menggunakan GDScript dengan menerapkan arsitektur *Component-Based Entity System* dan *Modular Design Pattern*.
3. Tahap Pasca-Produksi (*Post-Production*): Pelaksanaan pengujian fungsionalitas menggunakan *Black Box Testing*, verifikasi integrasi aset, debugging, serta eksekusi otomatisasi *build* melalui GitHub Actions.

Hubungan antarmetode dalam pipeline penelitian ini dapat dilihat pada Gambar E.1 dan Gambar E.2.



Gambar E.1 Diagram Alir Tahapan Penelitian



Gambar E.2 Diagram Alir Metodologi Penelitian

Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

2.2 Siklus Iterasi dan Kriteria Evaluasi

Proses iterasi dilakukan pada setiap penambahan fitur atau modul permainan melalui lima langkah berulang:

1. Design: Perancangan skenario logika (misalnya puzzle, AI musuh, atau interaksi UI) dan desain aset.
2. Implement: Pengodean logika pada GDScript dan integrasi aset 3D ke dalam scene Godot.
3. Test: Pelaksanaan pengujian fungsionalitas menggunakan Black Box Testing.
4. Evaluate: Evaluasi hasil pengujian berdasarkan kriteria keberhasilan yang telah ditetapkan. Jika terdapat bug, kegagalan logika, atau breakage pada build, proyek kembali ke tahap Design atau Implement.
5. Refine: Perbaikan kode (refactoring), penyesuaian parameter logika (seperti speed atau timer), serta optimasi aset.

Kriteria Evaluasi Penilaian:

1) Evaluasi Fungsionalitas Permainan: Modul dinyatakan Lolos/Berhasil apabila 100% langkah pengujian pada *Black Box Testing* menghasilkan luaran yang sesuai dengan ekspektasi skenario uji (seperti pintu terbuka setelah *double press*, *crosshair* dinamis merespons objek, dan FSM AI musuh bertransisi secara konsisten).
2) Evaluasi Pipeline CI**:** Workflow CI dinyatakan Lolos/Berhasil apabila eksekusi otomatisasi *build* menghasilkan status *Success* pada GitHub Actions dan menghasilkan artefak berkas *executable* (.exe) yang dapat dijalankan tanpa *error*.

2.3 Spesifikasi Perangkat Pengujian

Untuk menjamin keterulangannya (*replicability*), proses pengembangan, pengujian fungsional, dan eksekusi pipeline dilakukan pada lingkungan perangkat keras dan perangkat lunak berikut:

Spesifikasi Perangkat Keras (*Hardware*):

1) Processor (CPU): AMD Ryzen 5 7535HS
2) Memory (RAM): 16 GB DDR5
3) Graphics Processing Unit (GPU): NVIDIA GeForce RTX 2050 (VRAM 4 GB)
4) Penyimpanan (*Storage*): SSD 256 GB

Spesifikasi Perangkat Lunak (*Software*):

1) Game Engine: Godot Engine v4.5 (Standard Version)
2) Digital Content Creation (DCC) Tool: Blender v5.1.0
3) Format Integrasi Aset 3D: glTF 2.0 / GLB
4) Bahasa Pemrograman: GDScript
5) Version Control & CI Runner: Git & GitHub Actions

2.4 Prosedur Pengujian dan Kriteria Keberhasilan Pipeline CI

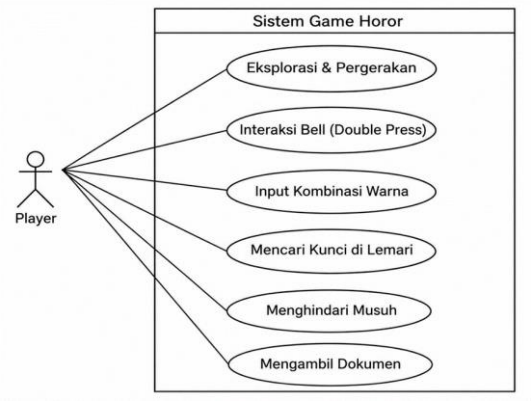
Pengujian fungsional (Black Box Testing) diawali dengan menyiapkan build executable atau menjalankan proyek di Godot Editor, mengeksekusi 13 skenario uji (seperti Main Menu, pergerakan, puzzle, AI musuh, UI, dan ending), lalu membandingkan hasilnya dengan expected output. Sementara itu, pipeline CI dijalankan otomatis melalui GitHub Actions saat pengembang melakukan git push ke main branch. Runner CI akan mengunduh repositori dan Godot Engine 4.5 beserta Export Templates, melakukan linting GDScript, mengompilasi proyek secara headless menjadi berkas .exe, lalu mengunggahnya sebagai Artifact Build. Pipeline CI dinyatakan berhasil jika seluruh skrip bebas error sintaksis (0 error), ekspor selesai dengan exit code 0, artefak .exe dan .pck dapat diunduh di GitHub Actions, serta berkas executable dapat dijalankan secara normal di lingkungan lokal.(Sutanto *et al.*, 2026)

Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

3. Hasil dan Pembahasan

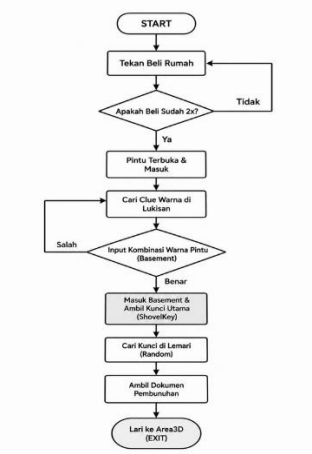
3.1 Hasil

Perancangan sistem game ini mengusung genre horror-stealth yang dipadukan dengan elemen teka-teki (puzzle). Seluruh bentuk interaksi pemain dengan sistem didokumentasikan dalam Use Case Diagram, yang mencakup fungsi eksplorasi lingkungan, interaksi tombol bel melalui mekanisme double press, penginputan kombinasi warna, pencarian kunci acak pada lemari, mekanisme penghindaran kecerdasan buatan (AI) musuh, hingga pengambilan dokumen sebagai syarat penyelesaian permainan.



Gambar 1 Use Case Diagram Sistem Game

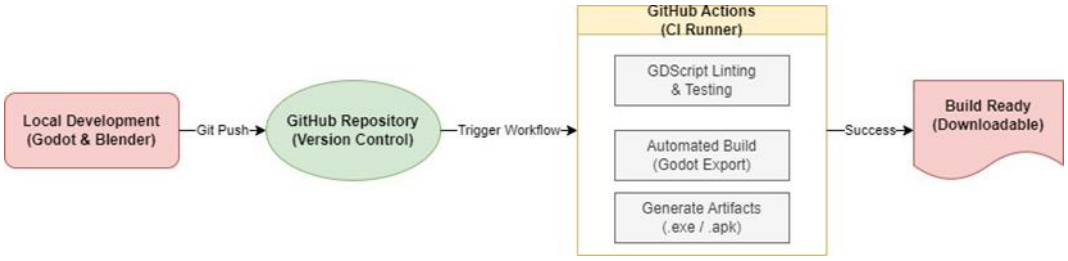
Alur logika permainan dirancang secara sistematis melalui urutan mekanik interaktif. Skenario diawali dari *Entrance Puzzle* yang memanfaatkan skrip pewaktu (*Timer*) untuk memvalidasi penekanan bel 2 kali dalam batas waktu 1,5 detik. Pemain kemudian diarahkan untuk mengidentifikasi petunjuk warna lukisan, melakukan pencarian lokasi kunci yang diacak menggunakan fungsi randi_range() di lemari *basement*, hingga mengeksekusi pemicuan *ending scene* secara otomatis ketika objek dokumen berhasil diambil.



Gambar 2 Flowchart Logika Permainan

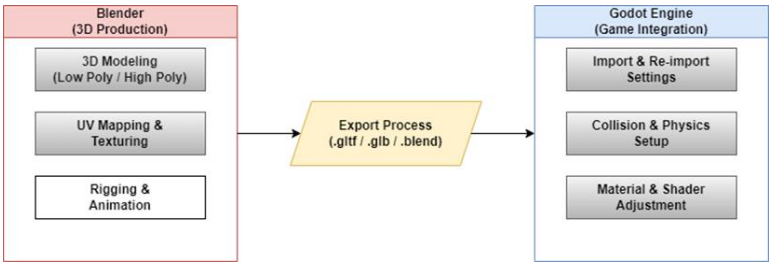
Arsitektur *Continuous Integration* (CI) dibangun dengan memanfaatkan platform GitHub Actions. Setiap kali pengembang melakukan *push* kode ke repositori GitHub, *workflow* CI secara otomatis mengeksekusi proses *linting* dan pengujian skrip GDScript, dilanjutkan dengan pengompilasi (*headless build*) proyek Godot menjadi berkas eksekusi (*executable* .exe) secara terotomatisasi.

Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152



Gambar 3 Arsitektur Pipeline Continuous Integration

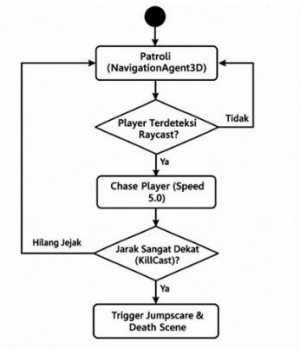
Pipeline aset 3D dari perangkat lunak Blender ke Godot Engine memanfaatkan format standar terbuka glTF 2.0. Tahapan alur kerja ini terdiri atas perancangan geometri, pemetaan UV (UV mapping), dan pemberian tekstur di Blender, dilanjutkan dengan ekspor berkas berformat .gltf/.glb, impor otomatis ke dalam Godot Engine, serta penyesuaian akhir berupa konfigurasi CollisionShape, material PBR (Physically Based Rendering), dan Shader.



Gambar 4 Pipeline Desain Aset 3D (Blender ke Godot Engine)

3.1.1 Implementasi Mekanik Game pada Godot 4.5

1) Sistem Interaksi Pemain: Diimplementasikan menggunakan RayCast3D pada kamera karakter. Fungsi check_for_interactable() mendeteksi metode seperti pickup_key(), toggle_door(), atau interact() pada objek sasaran serta mengubah modulasi warna crosshair secara real-time.
2) Puzzle Bell (Double Press): Menggunakan Timer 1.5 detik. Pintu terbuka hanya jika nilai press_count == 2 dicapai sebelum timer berakhir.
3) Randomized Key: Pengacakan posisi kunci dilakukan pada fungsi _ready() menggunakan randi_range() untuk memilih koordinat dari Array rand_positions.
4) AI Musuh (FSM): Musuh memanfaatkan NavigationAgent3D untuk patroli (kecepatan 3.0) dan beralih ke state chasing (kecepatan 5.0) saat RayCast3D mendeteksi pemain. Ketika kill_raycast bertabrakan dengan pemain, kamera beralih ke jumpscare_camera dan memicu transisi ke dead.tscn.



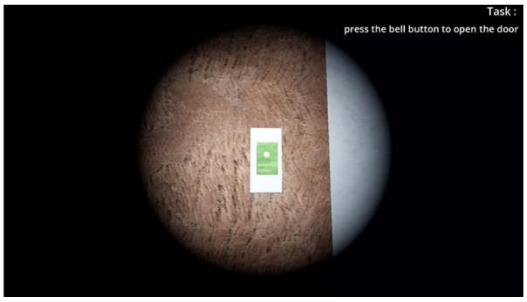
Gambar G.3 Activity Diagram Logika AI Musuh

Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

5). Puzzle Kombinasi Tombol: Validasi input array membandingkan urutan tombol dengan CORRECT_CODE = [1, 3, 2]. Batas waktu antar tekanan dikontrol oleh CODE_PRESS_WINDOW (2.0 detik).

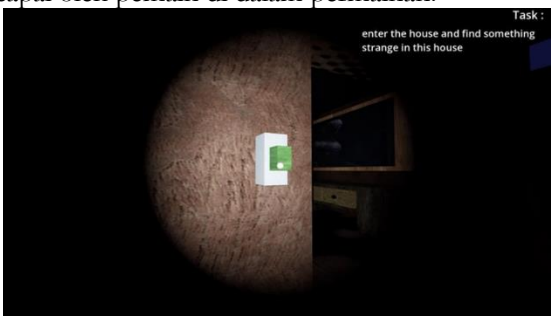
3.1.2 Implementasi Antarmuka (User Interface)

Penerapan *crosshair* dinamis dirancang secara khusus untuk menyembunyikan titik bidik pada layar secara *default* demi menciptakan tampilan visual yang bersih. Namun, titik bidik ini akan muncul secara otomatis dengan warna terang ketika pandangan pemain (*RayCast3D*) mengarah tepat ke objek atau elemen interaktif di dalam lingkungan permainan, sehingga interaksi tetap terasa intuitif tanpa mengurangi imersi horor yang dibangun.



Gambar 4 Implementasi Crosshair Dinamis saat Deteksi Objek.

Sistem *dynamic task messaging* diimplementasikan untuk menyajikan teks instruksi secara dinamis pada bagian atas antarmuka layar (HUD). Informasi instruksi—seperti panduan *"press the bell button to open the door"*—akan diperbarui secara otomatis dan *real-time* menyesuaikan tahapan serta progres penyelesaian misi yang dicapai oleh pemain di dalam permainan.



Gambar 5 Tampilan Instruksi Tugas Setelah Menyelesaikan Puzzle.

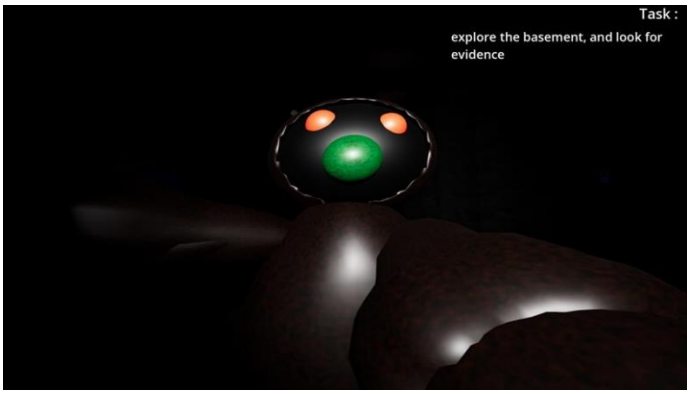
Fitur pencahayaan senter diimplementasikan memanfaatkan node SpotLight3D pada Godot Engine, yang dikonfigurasi menggunakan parameter pelemahan cahaya (attenuation) khusus. Konfigurasi ini menghasilkan efek penurunan intensitas cahaya (fall-off) yang realistis, sehingga pemain dapat menavigasi serta mengidentifikasi objek-objek penting secara presisi meskipun berada di dalam area ruang bawah tanah (basement) yang minim pencahayaan.



Gambar 6 Perbandingan Visual Ruangan dengan Aktivasi Senter.

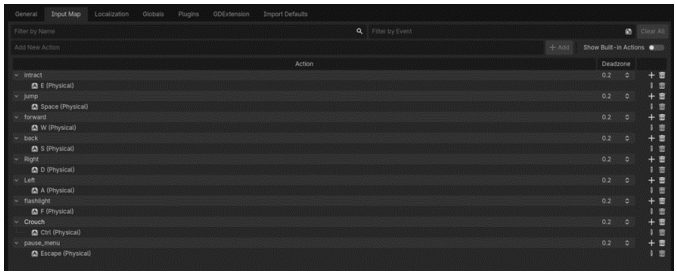
Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

Mekanisme *jumpscare* dan layar kematian (*death screen*) dirancang untuk mengambil alih kendali permainan dengan cara mengunci seluruh input pemain, memicu animasi *jumpscare* secara instan, serta mengeksekusi transisi efek karpet hitam (*fade-out*) otomatis menuju layar kematian atau penutup sesi permainan.



Gambar 7 Efek Visual Jumpscare dan Layar Akhir Permainan.

Seluruh pemetaan tombol dan kendali permainan dikonfigurasi melalui panel Input Map pada Project Settings Godot Engine. Setiap aksi fungsional dihubungkan dengan tombol fisik (Physical Key) untuk memastikan tata letak kontrol tetap konsisten, presisi, dan responsif terlepas dari variasi tata letak papan ketik (keyboard layout) yang digunakan oleh pemain.



Gambar 8 Panel Konfigurasi Input Map pada Editor Godot.

Tabel 1. Daftar Pemetaan Aksi dan Tombol Input

Action Name	Tombol Keyboard	Deskripsi Fungsi
Intract	E	Memicu fungsi interaksi pada Bell, Pintu, dan Puzzle.
Jump	Space	Memicu pergerakan vertikal (lompat) karakter.
forward	W	Menggerakkan karakter kearah depan.
Back	S	Menggerakkan karakter kearah belakang.
Right	D	Menggerakkan karakter kearah kanan.
Left	A	Menggerakkan karakter ke arah kiri.
Flashlight	F	Mengaktifkan atau mematikan sumber cahaya senter.
Crouch	Ctrl	Mengubah status pemain ke posisi crouch (jongkok).
Pause_menu	Esc	Membuka menu jeda permainan.

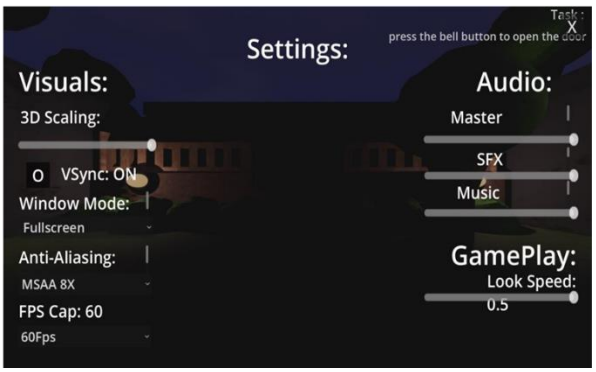
Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

Sistem menu secara keseluruhan dirancang secara terstruktur mulai dari *Main Menu* yang dilengkapi fitur dialog konfirmasi untuk mencegah kesalahan klik sebelum pemain masuk ke dalam permainan.



Gambar G.3.6 Implementasi Antarmuka Menu Utama.

Antarmuka *Pause Menu* diintegrasikan secara langsung dengan tampilan informasi tugas (*Task UI*) untuk meningkatkan efisiensi navigasi pemain. Selain itu, *Settings Menu* menyediakan opsi kustomisasi audio, pengaturan kontrol, dan preferensi grafis, yang datanya disimpan secara otomatis ke dalam berkas konfigurasi *.save_settings*.



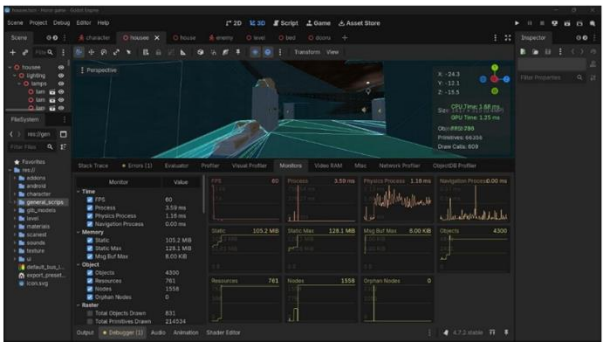
Gambar G.3.7 Tampilan Menu Pengaturan Audio dan Kontrol.

Seluruh alur permainan ditutup dengan *Ending Scene* berdurasi 6 detik yang memberikan transisi naratif secara dramatis sebelum secara otomatis mengembalikan tampilan layar ke menu utama.

3.1.3 Pengujian Performa Kuantitatif dan Pipeline CI

Untuk menjawab kebutuhan evaluasi performa terukur, dilakukan pengukuran metrik kinerja sistem saat permainan berjalan secara penuh (*runtime*) menggunakan *Monitor Profiler* bawaan Godot Engine 4.5 pada perangkat pengujian (AMD Ryzen 5 7535HS, RAM 16 GB, NVIDIA GeForce RTX 2050). Berdasarkan data metrik kinerja tersebut, permainan terbukti berjalan stabil pada batas 60 FPS dengan penggunaan memori statis sebesar 105,2 MiB dan *Static Max* mencapai 128,1 MiB. Visualisasi data hasil pemantauan performa sistem melalui *Monitor Profiler* Godot Engine disajikan pada Gambar di bawah ini.

Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152



Gambar G.3.8 Hasil Pemantauan Performa Sistem Menggunakan Monitor Profiler Godot Engine.

A. Metrik Kinerja Perangkat Lunak (Profiler Godot)

Berdasarkan data yang ditampilkan pada Gambar Monitor Profiler Godot, permainan menunjukkan kinerja yang stabil dengan tingkat bingkai (Frame Rate / FPS) mencapai batas 60 FPS. Waktu pemrosesan mencatat rata-rata Process Time sebesar 3,59 ms, sementara Physics Process Time tercatat pada 1,16 ms. Dalam konteks viewports, CPU Time berada di angka 1,68 ms dan GPU Time sebesar 1,25 ms. Penggunaan memori RAM (Static Memory) juga terpantau dengan baik, di mana penggunaan memori statis tercatat sebesar 105,2 MiB, dengan puncak penggunaan (Static Max) mencapai 128,1 MiB. Selain itu, manajemen objek dan komponen menunjukkan bahwa sistem dapat mengelola 1.558 Node, 761 Resource, 4.300 Object, serta 831 Total Objects Drawn, dengan total 214.534 Primitives. Terakhir, stabilitas memori terjaga dengan baik, terbukti tidak adanya kebocoran memori atau node terisolasi, yang ditunjukkan dengan nilai Orphan Nodes = 0.

B. Metrics Otomatisasi Build Continuous Integration (GitHub Actions)

Otomatisasi build yang dikonfigurasi melalui GitHub Actions terbukti mampu mengompilasi dan menghasilkan build ready-to-play secara stabil. Proses pengujian sintaksis GDScript dan ekspor headless Godot hingga menghasilkan artefak berkas .exe membutuhkan waktu rata-rata 2 menit 15 detik sejak pemicuan git push. Selain itu, tingkat keberhasilan seluruh eksekusi workflow pada repositori mencatatkan status Success, dengan tidak adanya kesalahan sintaks (0 syntax error).

3.1.4 Pengujian Fungsionalitas Sistem (Black Box Testing)

Pengujian fungsionalitas dilakukan melalui 13 skenario uji yang dirancang untuk memvalidasi alur utama permainan, logika puzzle, FSM AI musuh, serta antarmuka. Hasil perbaikan tabel pengujian (penyesuaian tata bahasa, konsistensi kolom, dan kejelasan uraian) disajikan pada Tabel 2:

Tabel 2. Hasil Pengujian Fungsionalitas Sistem (*Black Box Testing*)

No	Butir Uji	Langka Pengujian	Hasil Yang Diharapkan	Kesimpulan
1	Menu Utama	Menjalankan aplikasi dan klik tombol "Play".	Muncul panel Konfirmasi "Are You Sure".	Berhasil
2	Transisi Intro	Klik "Yes" pada panel konfirmasi.	Scene berpindah ke game_intro.tscn setelah 0.5 detik.	Berhasil
3	Gerakan Dasar	Menekan tombol W, A, S, D dan Space.	Karakter bisa berjalan dan melompat dengan lancar.	Berhasil
4	Tampilan Tugas	Mengamati Head-Up Display (HUD) di bagian atas layar.	Teks instruksi tugas ("press the bell button...")	Berhasil

Jurnal Ilmu Komputer dan Teknologi Informasi
https://journal.stmiki.ac.id
E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

			muncul secara dinamis	
5	Puzzle Bell	Menekan tombol 'E' pada objek Bell sebanyak 2 kali dalam jendela waktu 1,5 detik.	Suara bell berbunyi dan pintu utama terbuka.	Berhasil
6	Fitur Senter	Menekan tombol F saat di area gelap.	Lampu senter (SpotLight3D) menyala menerangi area pandang depan.	Berhasil
7	Menu Jeda	Menekan tombol 'Escape'saat permainan berlangsung.	Permainan berada pada posisi jeda (pause) dan antarmuka Pause Menu tampil.	Berhasil
8	Simpan Pengaturan	Mengubah nilai Volume/Sensitivitas lalu menutup panel Settings.	Data konfigurasi tersimpan secara otomatis ke dalam berkas .save_settings	Berhasil
9	Deteksi AI Musuh	Berjalan melintas di area garis pandang (*RayCast3D*) musuh.	AI musuh beralih dari status *Patrol* menjadi *Chasing* mengejar pemain.	Berhasil
10	Puzzle Kombinasi	Memasukkan kombinasi warna yang sesuai pada pintu *basement*.	Pintu *basement* terbuka (fungsi unlock_door aktif).	Berhasil
11	Kondisi Tertangkap	Karakter pemain terjangkau oleh kontak serangan AI musuh.	Pemicuan animasi *jumpscare* dan transisi otomatis kelayar kematian.	Berhasil
12	Penyelesaian Game	Mengambil berkas dokumen lalu melangkah menuju area *Exit*.	Tampilan layar menggelap secara perlahan (*fade-out*) dan beralih ke *scene Ending*.	Berhasil
13	*Restart Otomatis*	Menunggu tampilan layar *Ending* selama durasi 6 detik.	Sistem secara otomatis mengembalikan tampilan ke Menu Utama.	Berhasil

3.2 Pembahasan

Pembahasan mengenai otomatisasi build menggunakan GitHub Actions menunjukkan bahwa proses ini tidak hanya efisien, tetapi juga stabil dalam menghasilkan build yang siap dimainkan. Durasi rata-rata 2 menit 15 detik untuk menyelesaikan pengujian sintaksis GDScript dan ekspor headless Godot sejalan dengan penelitian oleh Faqih *et al.* (2024), yang menekankan pentingnya efisiensi waktu dalam pipeline pengembangan game. Tingkat keberhasilan yang mencapai status Success dengan tidak adanya kesalahan sintaks juga mencerminkan kualitas implementasi yang baik, yang sejalan dengan temuan Jonathan *et al.* (2023) mengenai pentingnya pengujian yang menyeluruh dalam pengembangan game untuk memastikan bahwa setiap komponen berfungsi dengan baik. Penelitian ini memberikan kontribusi signifikan terhadap literatur yang ada dengan menunjukkan bahwa otomatisasi build melalui GitHub Actions dapat diintegrasikan secara efektif dalam workflow pengembangan game 3D, mengurangi potensi kesalahan manusia dan meningkatkan produktivitas tim pengembang. Hasil ini juga mendukung argumen dari Martial *et al.* (2026) yang menyatakan bahwa integrasi alat otomatisasi dalam pipeline pengembangan dapat mempercepat proses pengembangan dan meningkatkan kolaborasi antar anggota tim. Dengan demikian, penelitian ini tidak hanya menambah wawasan tentang penggunaan GitHub Actions dalam pengembangan game, tetapi juga menunjukkan relevansi dan konsistensi dengan studi-studi sebelumnya yang membahas efisiensi dan efektivitas dalam pengembangan perangkat lunak.

E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

4. Kesimpulan dan Saran

Berdasarkan hasil penelitian dan pengembangan yang telah dilakukan, dapat disimpulkan bahwa penerapan metodologi Iterative Object-Oriented Development berhasil membangun pipeline pengembangan game 3D terpadu menggunakan Godot Engine 4.5. Integrasi aset visual dari Blender ke Godot Engine berjalan dengan lancar (seamless) menggunakan format standar terbuka glTF 2.0. Implementasi Component-Based Entity System dan Modular Design Pattern pada GDScript terbukti menghasilkan struktur kode yang terenkapsulasi, sehingga memudahkan proses pemeliharaan (maintainability) dan pengembangan modul fungsional (scalability). Hal ini dibuktikan oleh keberhasilan 100% pada 13 skenario pengujian fungsional yang dilakukan melalui metode Black Box Testing. Selain itu, konfigurasi Continuous Integration (CI) menggunakan GitHub Actions berhasil mengotomatisasi proses ekspor dan kompilasi build executable (.exe) secara stabil, dengan durasi rata-rata 2 menit 15 detik dan tingkat keberhasilan 100% tanpa adanya kesalahan sintaks (0 syntax error). Hasil pengujian performa menunjukkan stabilitas permainan pada tingkat bingkai 60 FPS, dengan alokasi memori statis rata-rata sebesar 105,2 MiB dan tanpa indikasi kebocoran memori, yang ditunjukkan dengan nilai Orphan Nodes = 0. Penelitian ini memberikan kontribusi penting dalam pengembangan game 3D, menunjukkan bahwa metodologi yang terstruktur dan penggunaan alat otomatisasi dapat secara signifikan meningkatkan efisiensi dan kualitas pengembangan.

5. Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Program Studi Teknologi Informasi Fakultas Teknik Universitas Muhammadiyah Palembang atas penyediaan fasilitas Laboratorium Komputer, serta seluruh pihak yang telah mendukung kelancaran penelitian ini.

6. Daftar Pustaka

Amir, M. F., Wardhono, W. S., & Arwani, I. (2017). Pengembangan Game 3D Warship Operation Trikora. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *1*(12), 1875-1882.

Del Gallego, N. P., Dacanay, M. J., & Valdecantos, A. V. (2026, June). Assessing the Build Accessibility of Open-Source Game Engines for Independent Game Development. In *2026 International Conference on Graphics and Signal Processing (ICGSP)* (pp. 21-28). IEEE.

Faqih, AR, Taufiqurrahman, A., Husen, JH, & Sabariah, MK (2024). Analisis empiris penggunaan alat CI/CD dalam alur kerja GitHub Actions. *Jurnal Informatika dan Rekayasa Web* , *3* (2), 251-261.

Guimarães, B., & Xexéo, G. (2026, September). A Declarative Model for Entity-Component-System Game Descriptions. In *Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)* (pp. 1004-1015). SBC. https://doi.org/10.5753/sbgames.2026.26347.

Hilmawan, B. N., & Yuniati, T. (2024). Perancangan Game Role-Playing sebagai Sarana Edukasi Sejarah Menggunakan Metode Game Development Life Cycle. *Computer Science (CO-SCIENCE)*, *4*(1), 1-10. https://doi.org/10.31294/coscience.v4i1.2032.

Johnson, J. (2023). *Buku Panduan Pengembangan Game Godot 4*. Packt Publishing.

E-ISSN: 3047-6674 | P-ISSN: 3047-6682
Vol. 4 No. 1 (2027) | Januari
OPEN ACCESS
https://doi.org/10.35870/jikti.v3i2.2152

Jonathan, M. W., Pragantha, J., & Haris, D. A. (2023). Pembuatan strategy game "Frontier of Hell" menggunakan Godot game engine. *Jurnal Serina Sains, Teknik Dan Kedokteran*, *1*(2), 393-400. https://doi.org/10.24912/jsstk.v1i2.31035.

Martial, L., Kubono, H., & Miura, S. (2026). From CityGML to Game Engine: A Reproducible Open-Source GIS Pipeline for Interactive 3D Urban Environments Using PLATEAU, Blender and Godot. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *50*, 143-149. https://doi.org/10.5194/isprs-archives-L-4-W1-2026-143-2026.

Mulachela, A., Rizki, K., & Wahyuddin, Y. A. (2020). Analisis Perkembangan Industri Game di Indonesia Melalui Pendekatan Rantai Nilai Global (Global Value Chain). *Indonesian Journal of Global Discourse*, *2*(2), 32-51. https://doi.org/10.29303/ijgd.v2i2.17.

Pratiwi, A. O., Qisthiano, M. R., & Aulia, N. A. (2025). Web Game Implementasi Metode Agile dalam Pembuatan Sistem Game Edukatif Berbasis Web untuk Pembelajaran Bahasa Inggris Tingkat Sekolat Dasar. *Jurnal Komtika (Komputasi dan Informatika)*, *9*(1), 100-111. https://doi.org/10.31603/komtika.v9i1.13495.

Sasongko, B. T. (2025, May). Pengembangan Game Escape From Basement dengan AI Berbasis Finite State Machine di Godot Engine Menggunakan Metode GDLC. In *Prosiding Seminar Nasional KONSTELASI* (Vol. 2, No. 1, pp. 95-106). https://doi.org/10.24002/prosidingkonstelasi.v2i1.11083.

Apriani, D., Darwis, M., & Putri, W. T. H. (2024). Pengembangan Game Fun Learning Untuk Siswa Sekolah Dasar Dengan Metode Game Development Life Cycle (GDLC). Jurnal Ilmu Komputer Dan Sistem Informasi (Jikomsi), 7(1), 238–245. https://doi.org/10.55338/jikomsi.v7i1.2919.

Fauzan, A., Harahap, N. S., Irsyad, M., & Pizaini, P. (2022). Pengembangan Game Action Sejarah Kerajaan Siak Sri Indrapura Bergenre Role Play Game. Jutisi Jurnal Ilmiah Teknik Informatika Dan Sistem Informasi, 11(3), 753. https://doi.org/10.35889/jutisi.v11i3.999.

Harsadi, P., Saptomo, W. L. Y., & Wardhana, C. Y. (2022). Implementasi Algoritma Fisher-Yates Shuffle Pada Game Edukasi Aksara Jawa Menggunakan Godot Engine. Jurnal Teknologi Informasi Dan Komunikasi (Tikomsin), 10(1). https://doi.org/10.30646/tikomsin.v10i1.603

Hidayat, M. S. (2023). Desain Property 3D Pada Metaverse Menggunakan Engine Unity untuk Medukung Platform IoT. Jurnal Ilmu Komputer Dan Bisnis, 14(2a), 43–50. https://doi.org/10.47927/jikb.v14i2a.524.

Rohman, M. N. A. A., Samodra, J., & Sutrisno, A. (2022). Desain Karakter 3D Game Anoman Obong untuk Remaja menggunakan Unity Render Toon Shading. Journal of Language Literature and Arts, 2(2), 151–167. https://doi.org/10.17977/um064v2i22022p151-167.

Sutanto, K., Lumba, E., No, J. L. R., & Ancol, P. (2026). PENGEMBANGAN GAME KUIS ARITMATIKA "MATH DUNGEON" BERBASISKAN METODE MULTIMEDIA DEVELOPMENT LIFE CYCLE. *matematika*, *6*, 7.

Wingqvist, D., Wickström, F., & Memeti, S. (2022, November). Evaluating the performance of object-oriented and data-oriented design with multi-threading in game development. In *2022 IEEE Games, Entertainment, Media Conference (GEM)* (pp. 1-6). IEEE.