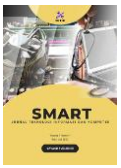


SMART : Jurnal Teknologi Informasi dan Komputer
Available online at https://journal.bengkuluinstitute.com/index.php/smart
Vol. 4 No. 2, July-December 2025, pages: 38-46
ISSN: 2963-6388
https://doi.org/10.58222/smart.v3i1.522

Implementation of a Finite State Machine for Character Animation Transitions in Unity-Based 3D Games

Implementasi Finite State Machine untuk Transisi Animasi Karakter pada Game 3D Berbasis Unity

Muhammad Fairul Filza [a]
Ahmad Zaid Rahman
Nadea Cipta Laksmita
Haryoko
Jedidta Adoni Saputra

Article history:

Submitted: 4 November 2022
Revised: 17 November 2022
Accepted: 28 November 2022

Keywords:

*F*SM, HSFM, Game

Abstract

A responsive character animation system with smooth transitions is a crucial factor in creating a high-quality and immersive gameplay experience. A common issue that frequently arises is animation that appears rigid or unnatural when a character switches from one action to another. This study aims to implement a Finite State Machine (FSM) to manage player character animation transitions in the game Legacy of the Sunstone. The method employed is a Hierarchical Finite State Machine (HFSM) with Moore Machine characteristics, in which the output in the form of animation is determined by the currently active state at a given time. The implementation was carried out using the Unity Engine and applied the State Pattern to establish a structured, modular, and maintainable code architecture. The CrossFadeInFixedTime technique was utilized to achieve smooth animation blending between states with configurable transition durations. The developed FSM system consists of two main categories: Locomotion states, including Idle, Movement, Jump, Falling, and Crouch, and Combat states, including Aim, Melee, and Takedown. System testing can be conducted using Black Box Testing through an iterative approach across 13 test scenarios that cover all functional requirements of the system.

Abstrak

Sistem animasi karakter yang responsif dan transisi yang mulus menjadi faktor krusial untuk menciptakan pengalaman bermain yang berkualitas dan imersif. Permasalahan yang sering muncul adalah animasi yang terlihat patah-patah atau tidak natural saat karakter berpindah dari satu aksi ke aksi lainnya. Penelitian ini bertujuan untuk menerapkan Finite State Machine dalam mengelola transisi animasi karakter pemain pada game Legacy of the Sunstone. Metode yang digunakan adalah Hierarchical Finite State Machine

[a] Universitas Amikom Yogyakarta

dengan karakteristik Moore Machine, di mana output berupa animasi ditentukan oleh state yang sedang aktif pada saat tertentu. Implementasi dilakukan menggunakan Unity Engine dan menerapkan State Pattern untuk menciptakan arsitektur kode yang terstruktur, modular, dan mudah dipelihara. Teknik CrossFadeInFixedTime digunakan untuk menghasilkan blending animasi yang mulus antar state dengan durasi transisi yang dapat dikonfigurasi. Sistem Finite State Machine yang dikembangkan mencakup dua kategori utama yaitu state Locomotion yang terdiri dari Idle, Movement, Jump, Falling, dan Crouch, serta state Combat yang terdiri dari Aim, Melee, dan Takedown. Pengujian dapat dilakukan dengan menggunakan metode Black Box Testing dengan pendekatan iteratif terhadap 13 skenario pengujian yang mencakup seluruh kebutuhan fungsional sistem.

SMART : Jurnal Teknologi Informasi dan Komputer © 2025.
This is an open access article under the CC BY-NC-SA license (https://creativecommons.org/licenses/by-nc-sa/4.0/).

Corresponding author:
Muhammad Fairul Filza
AMIKOM University
Email address: fairul.f@amikom.ac.id

1 Pendahuluan

Perkembangan game digital yang pesat telah meningkatkan tuntutan terhadap pengalaman bermain yang lebih imersif dan responsif. Dikutip dari berita internasional yang disusun oleh Lincoln Carpenter dan dipublikasikan pada awal Desember 2025, pasar game global diproyeksikan mencapai sekitar US$197 miliar pada akhir 2025, dengan pertumbuhan sekitar 7,5% terutama dari game PC dan mobile. Tren ini menunjukkan pertumbuhan pendapatan yang stabil dan besarnya kontribusi game terhadap industri hiburan global [1]. Salah satu elemen utama yang memengaruhi tingkat imersi pemain adalah kualitas animasi karakter, khususnya dalam skenario interaksi real-time. Animasi dalam game berkontribusi secara signifikan terhadap kualitas pengalaman bermain karena merupakan aspek kunci dalam menciptakan dunia game yang dapat dipercaya dan responsif terhadap input pemain [2].

Sistem animasi karakter yang responsif memerlukan real-time motion transition yang mampu menghasilkan perpindahan gerak secara cepat dan natural sesuai input pemain, sehingga tidak terjadi penundaan atau hentakan pada animasi [3]. Pengelolaan transisi animasi karakter pada sistem real-time menghadirkan berbagai tantangan teknis. Perpindahan antar aksi seperti berlari, melompat, menyerang, atau berjongkok sering kali menghasilkan gerakan yang tidak natural, perpindahan animasi yang mendadak, atau konflik antar state apabila ditangani menggunakan pendekatan logika kondisional konvensional. Sistem yang bergantung pada pemeriksaan kondisi langsung, seperti struktur if–else bertingkat, cenderung menjadi kompleks, sulit dipelihara, serta rentan terhadap kesalahan logika seiring bertambahnya jumlah aksi karakter.

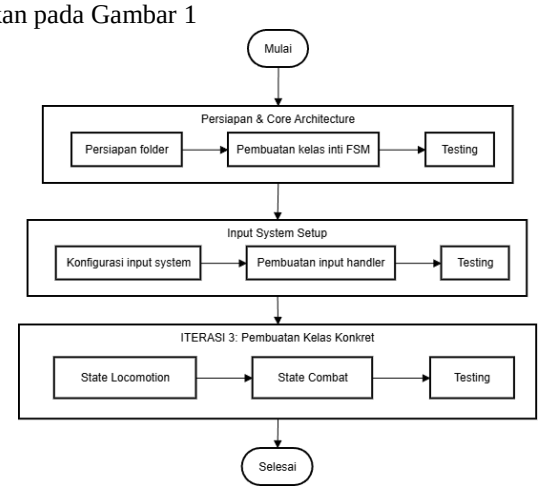
Permasalahan tersebut dapat diatasi dengan membuat mekanisme kontrol yang terstruktur guna mengatur perilaku animasi secara sistematis. Oleh karena itu, penelitian ini menerapkan pendekatan Hierarchical Finite State Machine (HFSM). Model HFSM yang digunakan dalam penelitian ini memiliki karakteristik Moore Machine, di mana keluaran sistem berupa animasi ditentukan sepenuhnya oleh state yang sedang aktif. Finite State Machine terdiri dari beberapa komponen fundamental yang bekerja secara terstruktur [4]. Komponen-komponen ini memastikan bahwa setiap transisi antar state bersifat deterministik dan dapat diprediksi berdasarkan input yang diberikan. Menurut NyStrom dan Rabin, sebuah Finite State Machine memiliki empat elemen utama yaitu state, transisition dan action [5], [6].

Moore Machine adalah jenis Finite State Machine di mana output hanya bergantung pada state saat ini, tanpa dipengaruhi oleh input [7]. Hal ini membuat output pada Moore Machine bersifat stabil dan dapat diprediksi karena output hanya berubah ketika terjadi perpindahan ke state baru. Kumar dan Singh dalam penelitiannya menjelaskan bahwa karakteristik ini menjadikan Moore Machine lebih mudah untuk dianalisis dan di-debug [7]. Contoh penerapan Moore Machine dalam game adalah sistem animasi sederhana, di mana animasi Idle selalu diputar selama karakter berada di state Idle, terlepas dari input yang diberikan.

Hierarchical Finite State Machine merupakan pengembangan dari Finite State Machine tradisional yang memungkinkan state memiliki sub-state di dalamnya [8], [6]. Hierarchical Finite State Machine memungkinkan pengembang untuk menyederhanakan proses komputasi dengan mengurangi kompleksitas transisi antar state [6].

2 Metodologi Penelitian

Metode *iteration* digunakan sebagai pipeline produksi untuk mengorganisir tahapan pembuatan komponen-komponen sistem secara sistematis. Pipeline produksi Finite State Machine pada game dirancang memiliki tiga iterasi utama sebagaimana ditunjukkan pada Gambar 1



Gambar 1. Pipeline Pengembangan Finite State Machine

3. Hasil dan Pembahasan

Hasil Analisis

Perancangan state dilakukan dengan mengidentifikasi seluruh perilaku karakter yang akan diatur melalui Finite State Machine. Pemetaan transisi dapat dilihat pada tabel 1 State Yang Dibutuhkan.

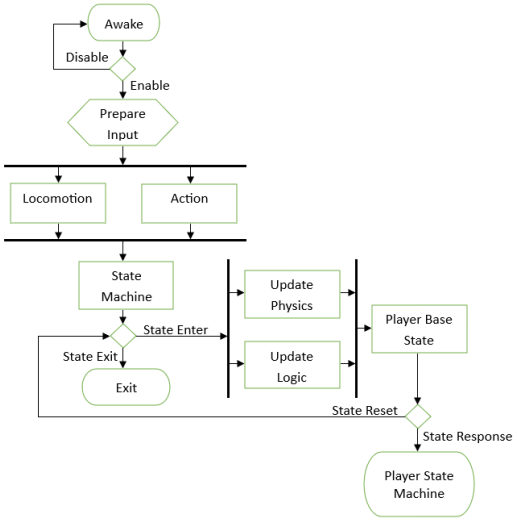
Tabel 1. State Yang Dibutuhkan

NO	Nama State
1	Aiming
2	Melee
3	Takedown
4	Crounch
5	Falling

6	Idle
7	Jump
8	Movement

Sistem input merupakan komponen penting dalam Finite State Machine yang berfungsi sebagai penghubung antara aksi pemain dengan transisi state. Pembuatan input menggunakan kelas InputHandler yang memanfaatkan Unity Input Sistem untuk mendeteksi dan memproses berbagai jenis input dari pemain.

Kelas State dibuat sebagai inti utama dari HFSM yang mana menerapkan prinsip enkapsulapsi dengan menyimpan referensi StateMachine secara protected. Objek yang diinialisasikan melalui konstruktor, memastikan setiap state memiliki akses terkontrol ke sistem pengelola utamanya. Struktur kelas ini mendefinisikan siklus hidup state melalui empat metode virtual yang dapat ditimpa oleh kelas turunan, yaitu: enter dan exit. Metode tersebut di eksekusi satu kali saat transisi terjadi untuk keperluan inisialisasi serta pembersihan, diikuti oleh UpdateLogic yang menangani logika permainan setiap frame, dan UpdatePhysics yang menangani kalkulasi fisika secara berkala. Pendekatan ini menerapkan konsep polymorphism, yang memungkinkan setiap perilaku spesifik karakter dikembangkan secara modular tanpa mengubah struktur inti sistem. Proses dari transisi dapat dilihat dari gambar 2 tentang diagram alur dari Finite State Machine.

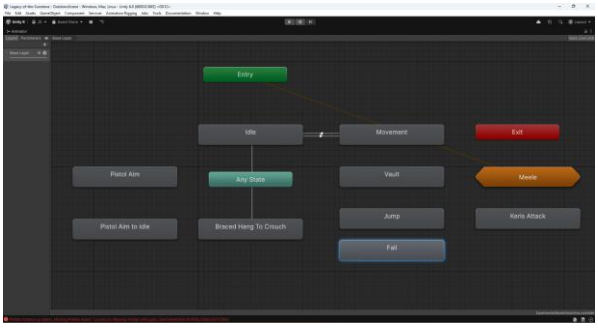


Gambar 2. Diagram Alur

Metode Awake mengumpulkan seluruh referensi komponen yang dibutuhkan, sedangkan metode Start menetapkan IdleState sebagai status awal karakter saat permainan dimulai. Proses StateMachine pada diagram diatas di atas dirancang mengelola pusat yang mewarisi MonoBehaviour, memungkinkannya untuk dipasang langsung pada objek karakter dalam lingkungan Unity. Fungsi inti dari kelas ini adalah metode ChangeState, yang menangani mekanisme transisi status secara sistematis dengan memastikan proses pembersihan pada status sebelumnya melalui pemanggilan metode Exit, sebelum menginisialisasi status baru melalui metode Enter. Selain mengatur transisi, kelas ini juga menerapkan pola delegasi pada siklus hidup permainan, di mana metode Update dan FixedUpdate secara terus-menerus menjalankan fungsi UpdateLogic dan UpdatePhysics milik status yang

sedang aktif (CurrentState), sehingga perilaku karakter dapat berubah secara dinamis sesuai dengan status yang dimilikinya saat itu.

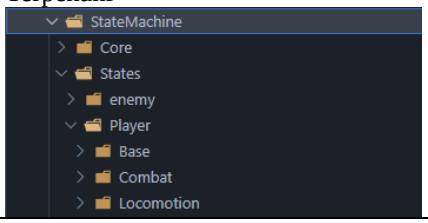
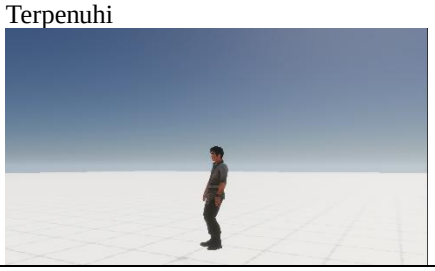
Setelah semua proses sudah dibuat, tahap selanjutnya adalah mengintegrasikan animasi-animasi dari masing masing state ke dalam Unity Animator. Berikut adalah tampilan Unity Animator yang telah dikonfigurasi dengan berbagai state animasi karakter pada Gambar 3 sebagai berikut.

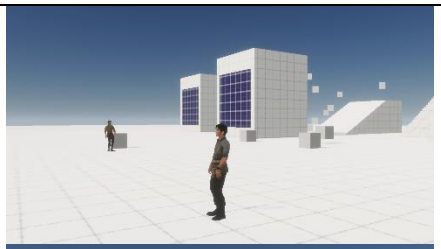
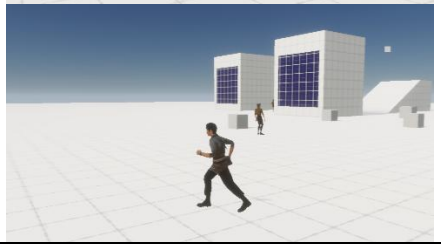
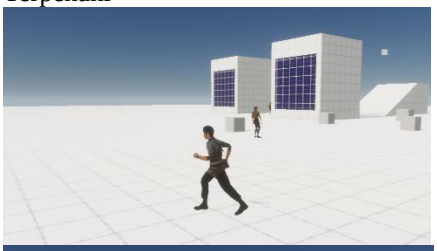
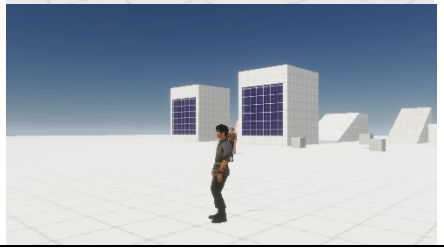
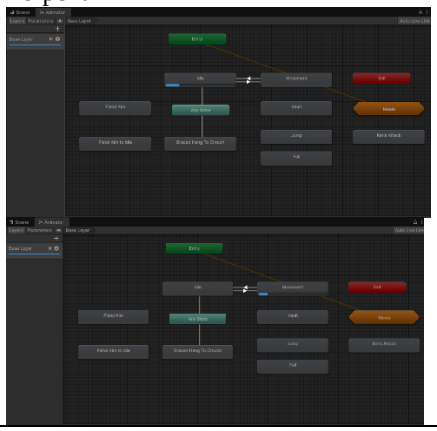


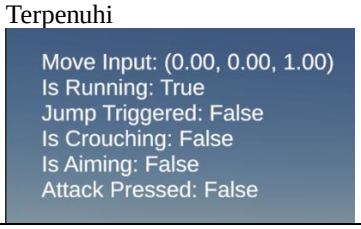
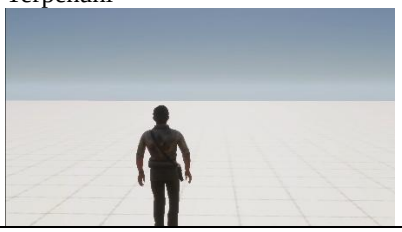
Gambar 3. Diagram Unity Animator

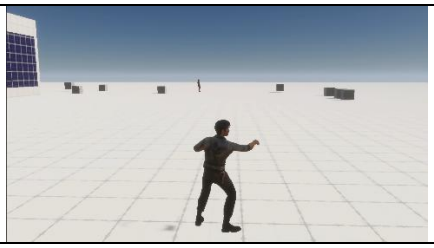
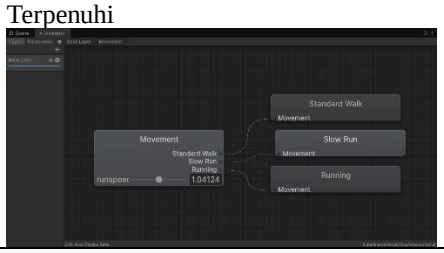
Pengujian kemudian dilakukan dengan melakukan black box testing. Pengujian ini akan menguji fungsionalitas tanpa melihat struktur kode internal program. Pengujian ini berfokus pada input yang diberikan dan output yang dihasilkan, memastikan bahwa sistem bekerja sesuai dengan spesifikasi kebutuhan yang ditentukan. Adapun hasil pengujian yang dilakukan secara independen terhadap setiap fitur utama pada Finite State Machine pada tabel 2 hasil pengujian.

Tabel 2. Hasil Pengujian

NO	Skenario Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil Pengujian
1	Pengujian struktur direktori.	Memeriksa struktur folder project di Unity.	Folder Core, States, Locomotion, dan Combat tersedia dan terorganisir dengan benar.	Terpenuhi 
2	Pengujian pengelolaan state karakter.	Memulai game dan mengamati state awal karakter.	Karakter berada pada state Idle dengan animasi Idle yang aktif.	Terpenuhi 
3	Pengujian transisi dari	Menekan tombol W/A/S/D saat	Karakter berpindah ke state Movement dengan animasi	Terpenuhi

	Idle ke Movement.	karakter dalam state Idle.	berjalan yang aktif.	 
4	Pengujian transisi dari Movement ke Idle.	Menekan tombol W/A/S/D saat karakter dalam state Idle.	Karakter berpindah ke state Idle dengan animasi Idle yang aktif	Terpenuhi  
5	Pengujian animasi sesuai state.	Mengamati animasi yang diputar pada setiap state.	Setiap state memainkan animasi yang sesuai seperti: Idle→Idle anim, Idle → Movement.	Terpenuhi 

6	Pengujian deteksi input pemain	Menekan berbagai tombol input (W, Shift, Space, Mouse).	Sistem mendeteksi dan mencatat status input pemain seperti Move Input, IsRunning, Jump Triggered, dan Attack Pressed	Terpenuhi 
7	Pengujian state Jump	Menekan tombol Space saat karakter grounded.	Karakter berpindah ke state Jump dengan animasi lompat yang aktif.	Terpenuhi 
8	Pengujian state Falling	Karakter jatuh dari ketinggian tanpa melompat.	Karakter berpindah ke state Falling dengan animasi jatuh yang aktif.	Terpenuhi 
9	Pengujian state Crouch	Menekan tombol Crouch saat karakter dalam state Idle/Movement.	Karakter berpindah ke state Crouch dengan animasi jongkok yang aktif.	Terpenuhi 
10	Pengujian state Aiming	Menekan tombol Aim saat karakter memiliki senjata.	Karakter berpindah ke state Aiming dengan animasi membidik yang aktif.	Terpenuhi 
11	Pengujian state Melee	Menekan tombol Attack saat karakter dalam state Idle/Movement.	Karakter berpindah ke state Melee dengan animasi serangan yang aktif.	Terpenuhi

				
12	Pengujian state Takedown.	Mendekati musuh dari belakang dan menekan tombol interaksi saat prompt muncul.	Karakter berpindah ke state Takedown dengan animasi eliminasi yang aktif.	Terpenuhi 
13	Pengujian transisi animasi yang mulus.	Melakukan transisi antar state secara berurutan.	Transisi animasi berjalan mulus tanpa perubahan mendadak menggunakan CrossFade.	Terpenuhi 

Berdasarkan keseluruhan hasil Black Box Testing pada Tabel 2, validasi black box menunjukkan bahwa setiap kebutuhan fungsional pada sistem Finite State Machine telah menghasilkan keluaran yang sesuai dengan hasil yang diharapkan. Pengujian pada 7 fitur utama yang mencakup struktur direktori, pengelolaan state karakter, animasi sesuai state, deteksi input pemain, state Locomotion, state Combat, dan transisi animasi yang mulus, seluruhnya memperoleh status sukses. Hasil ini membuktikan bahwa implementasi Finite State Machine berhasil mengatur transisi animasi karakter pemain secara efektif dan responsif sesuai dengan spesifikasi yang telah ditetapkan.

4 Kesimpulan dan Saran

Kesimpulan

Finite State Machine dapat digunakan untuk mengatur transisi animasi karakter pada game 3D dengan cara membagi perilaku karakter ke dalam state-state independen seperti Idle, Movement, Jump, Falling, Crouch, Aim, Melee, dan Takedown. Setiap state memiliki animasi yang sudah ditentukan dan akan diputar saat state tersebut aktif, sesuai dengan karakteristik Moore Machine di mana output animasi ditentukan oleh state yang sedang berjalan.

Implementasi Finite State Machine menggunakan arsitektur Hierarchical Finite State Machine dengan State Pattern pada Unity Engine. Hasil pengujian Black Box Testing terhadap 13 skenario menunjukkan bahwa seluruh fitur berfungsi sesuai harapan, membuktikan bahwa Finite State Machine mampu mengatur transisi animasi karakter secara efektif dan menghasilkan pengalaman bermain yang responsif.

Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa saran untuk pengembangan lebih lanjut:

1. Mengembangkan visual debugging tools pada Unity Editor untuk menampilkan state aktif dan transisi yang terjadi secara real-time guna mempermudah proses pengembangan.
2. Menambahkan unit testing untuk memastikan setiap state berfungsi dengan benar secara independen sebelum diintegrasikan.
3. Implementasi pada Enemy Artificial Intelegence Arsitektur Finite State Machine yang sama dapat diterapkan pada karakter musuh untuk mengelola perilaku Artificial Intelegence seperti Patrol, Chase, Attack, dan Retreat.

5 Daftar Pustaka

[1] Newzoo, "Gaming Industry Expected to Reach $197 Billion by 2025" PC Gamer, 2026. Diakses melalui https://www.pcgamer.com/gaming-industry/analyst-firm-says-the-games-market-is-expected-to-reach-a-record-breaking-usd197-billion-by-the-end-of-2025-driven-primarily-by-stronger-than-expected-performance-on-pc-and-mobile
[2] Wan Nor Raihan Wan Ramli, dkk, "The Needs in Developing Animation for Games Learning for Game Design Program", Open Access Journal Vol 14, Issue 3, (2025) E-ISSN: 2226-6348.
[3] Xiangjun Tang, dkk, ACM Transactions on Graphics (TOG), Volume 41, Issue 4 Article No.: 137, Pages 1 - 10
[4] M. Sipser, Introduction to the Theory of Computation, 2nd ed. Boston, MA: Thomson Course Technology, 2006.
[5] R. Nystrom, Game Programming Patterns. 2014.
[6] D. Jagdale, "Finite State Machine in Game Development," International Journal of Advanced Research in Science, Communication and Technology, pp. 384–390, Oct. 2021, doi: 10.48175/ijarsct-2062.
[7] R. Singh and G. Goyal, "Algorithm Design for Deterministic Finite Automata for a Given Regular Language with Prefix Strings," in SCRS CONFERENCE PROCEEDINGS ON INTELLIGENT SYSTEMS, Soft Computing Research Society, 2021, pp. 133–142. doi: 10.52458/978-93-91842-08-6-11.
[8] E. N. F. Dewi, A. N. Rachman, and R. H. Z. Hartadji, "Implementation of Hierarchical Finite State Machine for Controlling 2D Character Animation in Action Video Game," in 2023 8th International Conference on Informatics and Computing, ICIC 2023, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICIC60109.2023.10381978.
[9] M. Sipser, Introduction to the Theory of Computation, 2nd ed. Boston, MA: Thomson Course Technology, 2006.
[10] J. Schell, "The Art of Game Design: A Book of Lenses," 2019.
[11] R. Zubek, "Elements of Game Design," 2020.
[12] R. Nystrom, Game Programming Patterns. 2014.
[13] Unity Technologies, "Unity Engine: 2D & 3D Development Platform | Unity." Accessed: Dec. 19, 2025. [Online]. Available: https://unity.com/products/unity-engine
[14] Microsoft, "Visual Studio Code." Accessed: Dec. 19, 2025. [Online]. Available: https://code.visualstudio.com/docs/editor/whyvscode
[15] D. Farley, "Continuous Delivery Pipelines How to Build Better Software Faster," 2020. [Online]. Available: http://leanpub.com/cd-pipelines