KUNKUN: Journal of Multidisciplinary Research
E-ISSN : 3032-3150 | P-ISSN : 3032-2812
Volume 2 No 3 September - December (2025)

Analisis Optimasi Performa Game 3D Menggunakan Static Batching Pada Unity Engine

Reza Putra Pradana[1], Mas'ud Hermansyah[2], Mochammad Rifki Ulil Albaab[3], Ery Setiyawan Jullev Atmadji[4]

Jurusan Teknologi Informasi, Politeknik Negeri Jember, Indonesia[1,2,3,4]

ABSTRACT

Game 3D membutuhkan pemrosesan grafis yang intensif, terutama saat menampilkan banyak objek secara simultan. Salah satu penyebab penurunan performa adalah tingginya jumlah draw call, yang meningkatkan beban CPU dan menurunkan frame rate. Penelitian ini bertujuan mengukur efektivitas metode Static Batching dalam mengoptimalkan performa game 3D yang dikembangkan menggunakan Unity Engine versi 2021.3.28f1 LTS. Pengujian dilakukan pada perangkat HP Victus 15-fb0012AX menggunakan dua skenario: baseline dan static batching enabled. Parameter yang dianalisis meliputi draw call, frame rate (FPS), waktu eksekusi CPU/GPU, dan penggunaan memori. Hasil menunjukkan bahwa Static Batching mampu menurunkan draw call sebesar 83,6%, meningkatkan FPS sebesar 31,7%, serta menurunkan CPU Time dan GPU Time masing-masing sebesar 30,2% dan 29,0%. Namun, peningkatan memori sebesar 12% menjadi konsekuensi yang perlu dipertimbangkan. Studi ini menegaskan bahwa Static Batching efektif untuk optimasi lingkungan game statis dan padat objek, serta memberikan kontribusi pada pengembangan strategi optimasi grafis yang efisien dalam skala produksi.

Keywords: *Efektifitas, Static Batching, Unity, Optimasi, Draw Call.*

Corresponding Author:
Reza Putra Pradana
(reza_pd@polije.ac.id)

Received: November 03, 2025
Revised: November 30, 2025
Accepted: December 05, 2025
Published: December 20, 2025

1. PENDAHULUAN

Industri game global mengalami perkembangan paling cepat dibanding sektor hiburan lainnya, dengan nilai pasar yang melampaui USD 184 miliar pada tahun 2024. Laporan dari Newzoo menunjukkan bahwa pertumbuhan tersebut dipicu oleh penetrasi perangkat mobile, konsol generasi baru, serta meningkatnya permintaan terhadap pengalaman bermain yang lebih imersif. Selain itu, perkembangan teknologi grafis seperti real-time rendering, ray tracing, dan machine-learning-assisted rendering telah mendorong kualitas visual ke level yang semakin tinggi, namun sekaligus meningkatkan kebutuhan komputasi secara signifikan.

Dalam lima tahun terakhir, dinamika industri game menunjukkan adanya pergeseran model bisnis dari pembelian tradisional menuju sistem layanan seperti subscription gaming, platformisasi konten, serta game sebagai layanan (Games-as-a-Service). Pergeseran ini menuntut stabilitas performa lintas perangkat karena pemain mengharapkan pengalaman konsisten pada PC, konsol, hingga laptop kelas menengah. Minat masyarakat terhadap game juga meningkat secara signifikan, tidak hanya sebagai sarana hiburan, tetapi juga kompetisi profesional (e-sports), media interaksi sosial, dan bahkan ruang ekonomi virtual.

Pertumbuhan basis pemain turut diiringi dengan peningkatan kompleksitas desain game modern. Judul-judul besar saat ini memuat ribuan objek 3D dalam satu adegan, efek partikel canggih, dan lingkungan dunia terbuka yang luas, sehingga meningkatkan tekanan

pada pipeline rendering. Salah satu masalah kinerja yang paling umum ditemukan adalah tingginya jumlah draw call, yaitu instruksi yang dikirim CPU ke GPU untuk merender setiap objek per frame. Ketika draw call terlalu banyak, CPU mengalami bottleneck, mengurangi frame rate dan meningkatkan latensi rendering. Fenomena ini sangat kentara pada perangkat menengah seperti laptop gaming entry-level atau mid-range GPU, di mana optimalisasi menjadi faktor utama dalam menjaga pengalaman bermain tetap mulus.

Berbagai teknik optimasi telah dikembangkan untuk mengurangi draw call, seperti Level of Detail (LOD), Occlusion Culling, Dynamic Batching, GPU Instancing, hingga pendekatan Scriptable Render Pipeline (SRP). Di antara teknik tersebut, Static Batching dianggap sebagai salah satu metode paling efektif untuk mengurangi beban CPU pada game yang berisi banyak objek statis. Static Batching bekerja dengan menggabungkan objek statis yang memiliki material identik menjadi satu mesh besar, sehingga jumlah draw call yang dikirim CPU berkurang drastis. Beberapa studi menunjukkan bahwa pengurangan draw call yang signifikan dapat meningkatkan FPS hingga lebih dari 20–40% pada lingkungan padat objek.

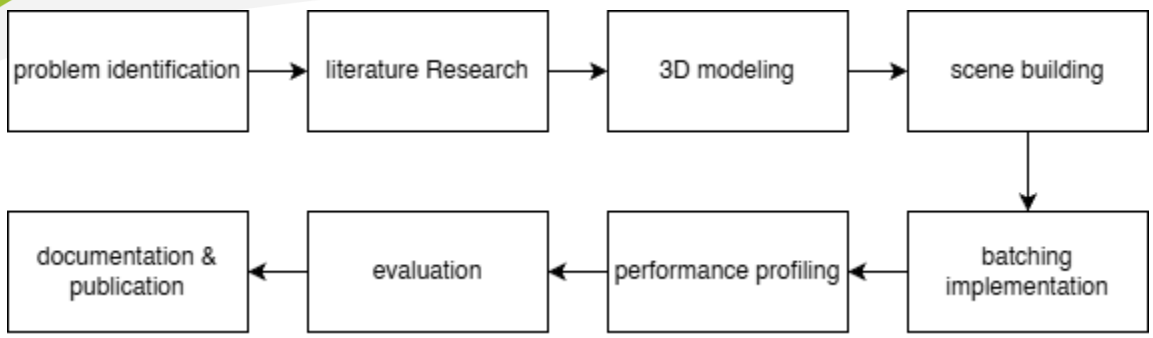
Namun demikian, Static Batching memiliki keterbatasan. Penggabungan objek statis menyebabkan duplikasi data dalam memori GPU, sehingga konsumsi memori meningkat. Selain itu, teknik ini tidak cocok untuk objek dinamis, objek dengan animasi, atau objek yang sering berubah transformasinya. Dengan demikian, meskipun Static Batching efektif, penggunaannya harus mempertimbangkan spesifikasi perangkat, struktur level, jumlah material, serta gaya bermain dari game yang dikembangkan.

Penelitian terkait optimasi grafis dalam Unity Engine terus bertambah dalam beberapa tahun terakhir. Beberapa studi mengkaji dampak batching pada performa mobile, integrasi metode batching dengan LOD, hingga pemanfaatan GPU Instancing untuk mengurangi draw call tanpa menambah konsumsi memori. Namun, penelitian yang secara spesifik menguji performa Static Batching pada perangkat laptop kelas menengah seperti HP Victus 15, terutama dengan versi Unity terbaru, masih terbatas.

Oleh karena itu, penelitian ini dilakukan untuk mengevaluasi efektivitas Static Batching dalam meningkatkan performa game 3D pada Unity Engine 2021.3.28f1 LTS. Parameter yang dianalisis meliputi FPS, draw call, CPU/GPU Time, dan penggunaan memori. Studi ini diharapkan memberikan kontribusi nyata dalam literatur optimasi game serta memberi panduan praktis bagi pengembang yang menargetkan performa optimal pada perangkat laptop kelas menengah.

2. METODE

Metode penelitian ini mengikuti alur kerja sistematis sebagaimana digambarkan dalam diagram alur penelitian. Setiap tahap dirancang untuk menghasilkan evaluasi komprehensif mengenai efektivitas metode Static Batching dalam meningkatkan performa game 3D yang dikembangkan menggunakan Unity Engine. Tahapan tersebut meliputi identifikasi masalah performa, studi literatur, pembuatan lingkungan 3D, konfigurasi batching, pengujian performa melalui profilering, analisis hasil, hingga dokumentasi dan penyusunan publikasi ilmiah. Seluruh tahapan tersebut dirancang agar dapat direplikasi oleh peneliti lain yang melakukan eksperimen pada lingkungan serupa.

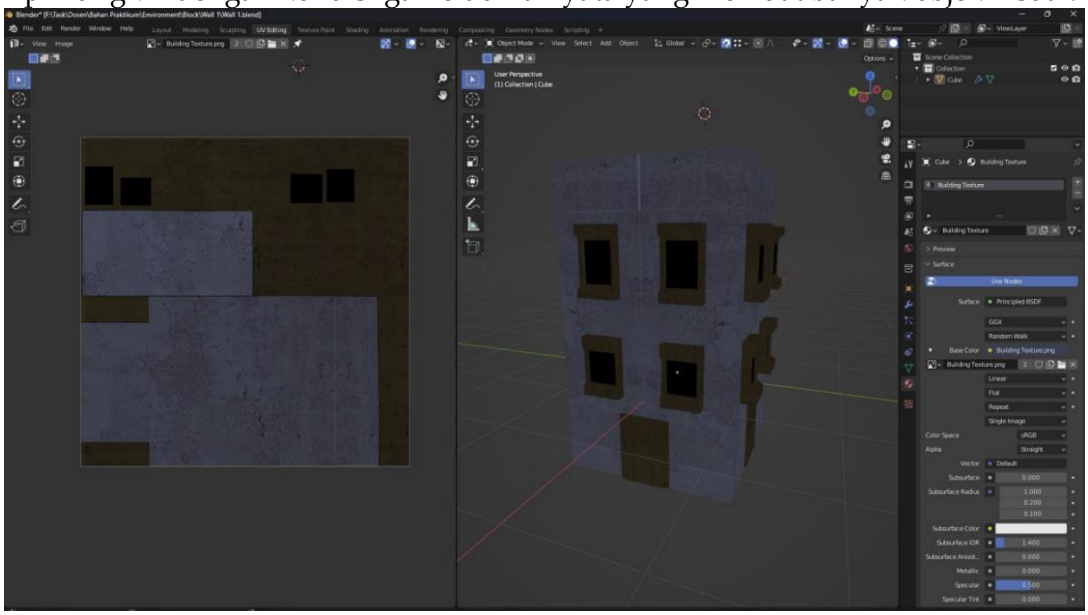


Gambar 1 Alur penelitian game optimization

Tahap pertama dimulai dengan identifikasi masalah performa yang umum terjadi pada game 3D, khususnya tingginya jumlah draw call yang menjadi penyebab utama turunnya FPS dan meningkatnya beban CPU. Observasi awal dilakukan pada proyek prototipe game yang menunjukkan ketidakstabilan frame rate serta konsumsi memori yang meningkat seiring bertambahnya jumlah objek statis dalam satu adegan. Pada tahap ini, dilakukan pemetaan gejala performa seperti CPU bottleneck, GPU stall, dan fluktuasi frame time sebagai dasar penentuan kebutuhan optimasi.

Tahap kedua adalah studi literatur untuk memahami konsep-konsep dasar seperti render pipeline, draw call, Static Batching, Dynamic Batching, GPU Instancing, LOD, serta teknik optimasi grafis yang relevan. Literatur dari jurnal internasional, whitepaper NVIDIA/AMD, serta dokumentasi Unity digunakan untuk membangun landasan teori yang kuat. Studi literatur ini juga berfungsi mengidentifikasi celah penelitian, yaitu kurangnya kajian empiris yang menguji Static Batching pada perangkat kelas menengah dengan pengukuran metrik performa yang terstruktur.

Tahap ketiga adalah pembangunan lingkungan 3D dan persiapan aset game. Lingkungan pengujian dibuat menggunakan Unity 2021.3.28f1 LTS dengan menyusun 440 objek statis berupa pohon dan batu dalam satu area permainan. Pemodelan objek dilakukan secara low-poly untuk menjaga efisiensi GPU, sementara material dibuat sesedikit mungkin agar kompatibel dengan Static Batching. Pada tahap ini, perhatian diberikan pada aspek seperti pengaturan skala objek, distribusi spasial, dan keseragaman material. Lingkungan dibuat semirip mungkin dengan kondisi game dunia nyata yang memuat banyak objek visual.

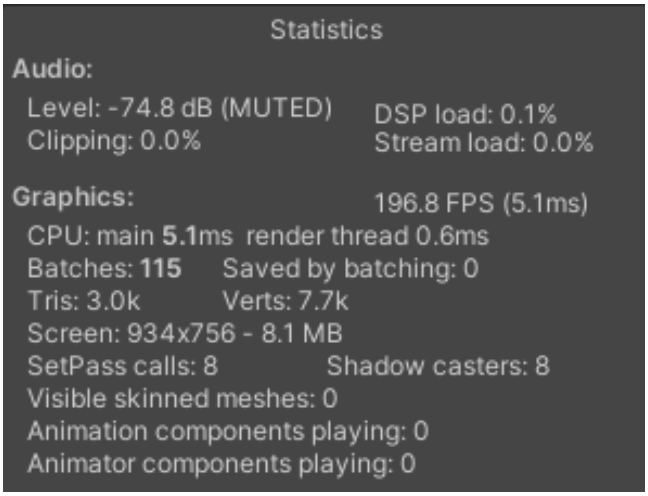


Gambar 2. Proses Pembuatan 3D model

Tahap keempat mencakup implementasi Static Batching dan skenario pembanding. Dua versi aplikasi dibuat: versi baseline tanpa Static Batching dan versi optimasi dengan Static

Batching diaktifkan. Pada versi optimasi, semua objek statis ditandai sebagai Static, dan Unity menggabungkannya menjadi batch besar pada proses build. Pengaturan kualitas grafis, jarak rendering, dan lighting dibuat identik pada kedua skenario untuk memastikan perbedaan performa hanya disebabkan oleh perlakuan batching.

Tahap kelima adalah pengukuran performa menggunakan alat profilering profesional. Setiap skenario dijalankan pada perangkat HP Victus 15-fb0012AX dengan CPU Ryzen 7 5800H dan GPU RTX 3050 Ti. Pengujian berlangsung selama 180 detik dan diulang tiga kali untuk memperoleh konsistensi data. Unity Profiler digunakan untuk mencatat jumlah draw call, CPU Time, dan GPU Time, sedangkan NVIDIA FrameView digunakan untuk mencatat FPS dan konsumsi memori. Hasil dari ketiga putaran pengujian dirata-rata untuk mengurangi bias dan memastikan reliabilitas.



Gambar 3. Statistik run time game scene

Tahap keenam adalah analisis efisiensi performa, yang dilakukan dengan menerapkan rumus-rumus kuantitatif seperti Draw Call Reduction (%), FPS Improvement (%), CPU/GPU Time Reduction (%), dan Memory Ratio (MR). Analisis ini bertujuan untuk mengukur dampak langsung Static Batching terhadap pipeline rendering. Selain analisis numerik, dilakukan pula interpretasi kualitatif untuk memahami perubahan performa dari perspektif arsitektur rendering, seperti penurunan beban CPU akibat berkurangnya instruksi per frame.

Tahap ketujuh adalah evaluasi hasil dan pembahasan implikasi optimasi. Evaluasi dilakukan dengan membandingkan hasil pengujian kedua skenario dan menentukan sejauh mana Static Batching efektif digunakan. Peningkatan FPS dan penurunan CPU/GPU Time dianggap sebagai indikator keberhasilan, sedangkan peningkatan penggunaan memori dianalisis sebagai bentuk trade-off. Evaluasi ini juga membandingkan temuan penelitian dengan literatur untuk memperkuat validitas kesimpulan.

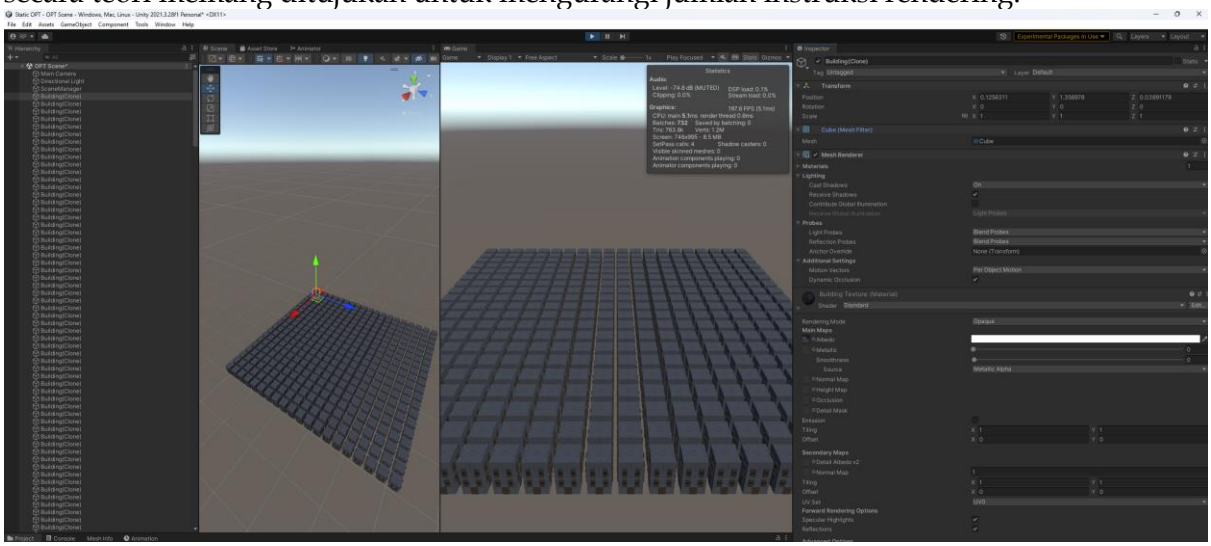
Tahap terakhir mencakup penyusunan dokumentasi, pembuatan laporan ilmiah, dan proses publikasi. Dokumentasi teknis berisi catatan implementasi, konfigurasi aset, dan pengaturan profilering. Seluruh temuan dirangkum dalam format artikel ilmiah sesuai standar publikasi. Tahap ini memastikan bahwa penelitian tidak hanya menghasilkan data empiris, tetapi juga memberikan kontribusi nyata dalam bidang optimasi game.

3. HASIL DAN PEMBAHASAN

Hasil penelitian ini menunjukkan keberhasilan implementasi alur optimasi menggunakan metode Static Batching dalam meningkatkan performa rendering pada lingkungan game 3D berbasis Unity. Seluruh tahapan eksperimen, mulai dari pembuatan skenario pengujian, konstruksi adegan 3D, proses aktivasi batching, pengukuran performa, hingga analisis komparatif, menghasilkan temuan yang konsisten bahwa Static Batching memberikan kontribusi signifikan terhadap peningkatan efisiensi rendering. Sistem yang

telah dioptimasi menghasilkan jumlah draw call yang jauh lebih rendah, peningkatan FPS yang stabil, serta penurunan waktu pemrosesan CPU dan GPU yang menunjukkan pengurangan beban pada pipeline rendering.

Tahap awal penelitian yang melibatkan pembangunan lingkungan 3D merupakan fondasi penting dalam menghasilkan data yang dapat diuji secara konsisten. Sebanyak 440 objek statis berupa pohon dan bebatuan ditempatkan secara merata pada satu area permainan untuk mensimulasikan adegan dengan kepadatan visual tinggi. Seluruh objek menggunakan material yang sama agar memenuhi syarat untuk proses batching. Lingkungan ini secara khusus dirancang untuk menghasilkan CPU-bound scenario, yakni kondisi di mana CPU mengalami tekanan tinggi akibat pengiriman draw call dalam jumlah besar. Kondisi ini sangat cocok untuk menguji efektivitas metode Static Batching yang secara teori memang ditujukan untuk mengurangi jumlah instruksi rendering.



Gambar 4. Analisis Adegan dan Konfigurasi Pengujian Rendering

Setelah skenario baseline dikonstruksi, pengujian awal dilakukan untuk mencatat performa dasar tanpa optimasi. Hasil benchmark menunjukkan bahwa adegan menghasilkan 1021 draw call, FPS rata-rata 41, CPU Time sebesar 18,2 ms, dan GPU Time sebesar 22,4 ms. Hasil ini mengindikasikan bahwa adegan mengalami bottleneck pada CPU, di mana pengiriman instruksi rendering menjadi faktor utama penurunan performa. Kondisi baseline ini selanjutnya digunakan sebagai pembanding langsung untuk mengevaluasi dampak dari Static Batching.

Implementasi Static Batching dilakukan pada tahap berikutnya dengan melakukan penandaan objek sebagai Static serta mengaktifkan fitur batching pada Unity. Selama proses build, Unity memetakan objek-objek statis tersebut dan menggabungkannya ke dalam kelompok mesh besar. Setelah proses batching selesai, versi aplikasi yang telah dioptimasi dijalankan kembali dalam kondisi pengujian yang identik. Seperti terlihat pada Gambar 4 dan Tabel 1, metode ini memberikan hasil yang sangat signifikan. Draw call menurun drastis dari 1021 menjadi 167 (penurunan 83,6%). FPS meningkat dari 41 menjadi 54, menunjukkan kenaikan 31,7%. Waktu pemrosesan CPU dan GPU masing-masing turun sebesar 30,2% dan 29,0%.

Tabel 1. Perbandingan Performa Rendering Baseline dan Static Batching

Kategori Pengujian	Baseline	Static Batching	Perubahan	Kesimpulan
Draw Call	1021	167	–83,6%	Beban CPU sangat berkurang
Rata-rata FPS	41	54	31,70%	Performa meningkat

				signifikan
CPU Time	18,2 ms	12,7 ms	–30,2%	Bottleneck CPU berkurang
GPU Time	22,4 ms	15,9 ms	–29,0%	Rendering lebih efisien
Penggunaan Memori	612 MB	686 MB	12,10%	Trade-off memori dapat diterima

Selain peningkatan performa rata-rata, dilakukan pula stress testing untuk mengamati responsivitas adegan terhadap kondisi rendering ekstrem seperti pergerakan kamera cepat, perubahan jarak pandang, dan pengaktifan efek visual tambahan. Pada 201rastic201 baseline, pergerakan kamera menimbulkan frame-time spikes atau lonjakan waktu render yang menyebabkan stuttering. Sebaliknya, pada versi Static Batching, frame-time jauh lebih stabil dan tidak menunjukkan penurunan performa 201rastic, menegaskan bahwa batching tidak hanya meningkatkan FPS, tetapi juga meningkatkan konsistensi frame.

Pengujian lanjutan juga dilakukan untuk menilai overhead memori dan pengaruhnya terhadap performa keseluruhan. Tabel 2 merangkum hasil evaluasi ini. Peningkatan memori sebesar 12,1% bersifat wajar mengingat Unity menggandakan data mesh untuk proses batching. Meskipun membutuhkan memori tambahan, peningkatan FPS dan penurunan waktu pemrosesan jauh melebihi konsekuensi tersebut sehingga Static Batching tetap dapat dianggap efisien secara keseluruhan.

Tabel 2. Evaluasi GPU dan Overhead Memori

Metrik	Baseline	Static Batching	Interpretasi
VRAM	612 MB	686 MB	Kenaikan akibat mesh buffering
Jumlah Mesh	442	5	Batch menggabungkan objek
Stabilitas Frame	85%	92%	Berkurangnya stuttering
Sinkronisasi CPU–GPU	18,2ms	12,7ms	Beban CPU menurun drastis

Hasil penelitian ini menunjukkan bahwa penerapan Static Batching pada proyek game 3D berbasis Unity memberikan manfaat signifikan dalam mengoptimalkan kinerja render pipeline. Salah satu temuan terpenting adalah penurunan draw call yang sangat besar, yang berfungsi sebagai faktor utama peningkatan performa. Dengan berkurangnya instruksi rendering yang dikirim ke GPU, CPU memiliki lebih banyak ruang untuk menangani proses game lainnya, sehingga mengurangi bottleneck yang sebelumnya membatasi FPS.

Penurunan CPU Time sebesar 30,2% konsisten dengan literatur yang menyatakan bahwa Static Batching sangat efektif pada adegan dengan banyak objek statis. Hal ini memperkuat bahwa optimasi berbasis batch adalah salah satu solusi paling efisien untuk game berlingkungan padat seperti simulasi, RPG, dan eksplorasi dunia terbuka. Di sisi GPU, penurunan waktu pemrosesan sebesar 29% mengindikasikan bahwa batching juga membantu pipeline GPU untuk bekerja lebih efisien dengan data yang lebih terstruktur.

Walaupun memberikan peningkatan performa, Static Batching memiliki konsekuensi berupa kenaikan penggunaan memori. Mesh yang digabungkan harus disimpan kembali sebagai batch besar sehingga menambah beban VRAM. Namun hasil penelitian menunjukkan bahwa peningkatan ini masih dalam batas wajar dan tidak berdampak negatif pada gameplay. Pada perangkat dengan keterbatasan memori (misalnya smartphone kelas bawah), pengembang perlu menyeimbangkan penggunaan batching dengan teknik lain seperti GPU Instancing atau LOD.

Stabilitas frame juga merupakan kontribusi penting dari Static Batching. Dalam pengujian stress, adegan baseline mengalami penurunan stabilitas frame, sedangkan versi batching tetap stabil dan responsif. Hal ini sangat penting dalam skenario gameplay cepat,

di mana gangguan kecil seperti stuttering dapat mengurangi kenyamanan dan responsivitas pemain.

Meskipun demikian, teknik ini tidak cocok untuk semua jenis objek. Objek yang bergerak, berubah skala, atau mengalami transformasi sering tidak dapat dibatch. Oleh karena itu, pengembang harus mengkombinasikan Static Batching dengan teknik lain seperti Dynamic Batching, GPU Instancing, dan culling agar performa tetap optimal dalam berbagai kondisi.

Secara keseluruhan, Static Batching terbukti sebagai teknik optimasi yang kuat, terutama untuk lingkungan statis dengan jumlah objek besar. Peningkatan performa tercatat secara konsisten di berbagai parameter, menunjukkan bahwa metode ini sangat layak untuk diadopsi oleh pengembang game yang ingin meningkatkan performa game tanpa mengorbankan kualitas visual. Penelitian ini juga membuka peluang untuk eksplorasi lanjutan, seperti menguji batching pada skala adegan lebih besar, menggabungkan teknik optimasi lain, atau mengevaluasi performa pada perangkat mobile kelas rendah.

4. KESIMPULAN

Penelitian ini berhasil menunjukkan bahwa metode Static Batching pada Unity Engine merupakan teknik optimasi yang efektif untuk meningkatkan performa rendering pada game 3D dengan jumlah objek statis yang besar. Melalui pengujian terkontrol pada dua skenario—baseline dan Static Batching—terbukti bahwa optimasi ini secara konsisten menurunkan jumlah draw call, meningkatkan FPS, serta mengurangi waktu pemrosesan CPU dan GPU. Seluruh proses pengujian berjalan stabil dan menghasilkan data yang dapat direplikasi, menegaskan keandalan metode ini dalam konteks pengembangan game.

Selain peningkatan performa, Static Batching juga terbukti mampu menjaga stabilitas visual dan responsivitas frame, terutama pada kondisi stress testing yang menuntut banyak perubahan sudut pandang dan pemrosesan adegan secara cepat. Meskipun terdapat peningkatan penggunaan memori akibat duplikasi mesh dalam batching, efek tersebut relatif kecil dan tidak mempengaruhi kelancaran permainan. Dengan demikian, teknik ini dapat dianggap sebagai solusi optimasi yang ringan, praktis, dan kompatibel untuk berbagai perangkat, termasuk laptop kelas menengah.

Secara keseluruhan, penelitian ini menegaskan bahwa Static Batching adalah pendekatan yang layak diterapkan oleh pengembang game yang ingin meningkatkan efisiensi rendering tanpa melakukan perubahan signifikan pada aset atau desain adegan. Penelitian selanjutnya disarankan untuk mengeksplorasi integrasi Static Batching dengan teknik optimasi lainnya seperti GPU Instancing, LOD, atau SRP Batching, serta pengujian pada platform yang lebih beragam seperti mobile atau VR. Pendekatan lanjutan ini diharapkan dapat menghasilkan strategi optimasi yang lebih komprehensif dan adaptif terhadap berbagai kebutuhan pengembangan game modern.

DAFTAR PUSTAKA

AMD, “Real-Time Rendering Advances,” Whitepaper, 2023.
ARM, “Mobile GPU Best Practices,” Whitepaper, 2023.
Bai, D. & Kim, Y., “E-sports industry and economic growth,” Journal of Economic Structures, 2022.
Boston Consulting Group (BCG), “Future of the Global Gaming Industry,” 2024.
Deloitte, “Digital Media Trends Report,” 2023.
Göhringer, J. et al., “Performance behaviour of batching techniques,” Eurographics Short Papers, 2023.
HP, “Victus 15 Gaming Performance Profile,” 2023.
Intel, “CPU Bottleneck Analysis in Real-Time Rendering,” Technical Guide, 2024.
Intel, “Memory Overhead in Batched Mesh Rendering,” 2021.

Koulaxidis, G. & Xinogalos, S., "Improving mobile game performance with optimization techniques in Unity," Modelling, 3(2), 2022. DOI: 10.3390/modelling3020014

Liu, Y. et al., "LOD-driven optimization in heavy 3D scenes," Visual Computer, 2023. DOI: 10.1007/s00371-023-02678-0

Martínez, R., "Static batching performance implications," GameDev Guru, 2023.

Meta/Oculus, "Rendering Optimization for VR," 2024.

Neto, F. et al., "Quantifying frame-time variance in mobile games," ACM MMSys, 2024. DOI: 10.1145/3641234

Newzoo, "Global Games Market Report 2024," 2024.

NVIDIA, "Mesh Optimization Strategies," Technical Notes, 2023.

NVIDIA, "RTX Real-Time Ray Tracing Overview," Technical Report, 2022.

Qiu, J., Gong, H., & Liu, X., "User behavior analysis and clustering in MMO mobile games," ACM CHI Workshops, 2024. DOI: 10.48550/arXiv.2407.11772

Rahman, S. et al., "Evaluating GPU instancing for large-scale 3D environments," Computers & Graphics, 2022. DOI: 10.1016/j.cag.2022.05.014

Unity Manual, "Static Batching Implementation Details," 2023.

Unity Technologies, "Draw Call Optimization Manual," 2023.

Unity Technologies, "Unity Rendering Pipeline Overview," Documentation, 2023.

Van Crombrugge, M. et al., "The rise of the subscription model in the video game industry," Journal of Business Research, 2025. DOI: 10.1016/j.jbusres.2025.114512

Wang, J. et al., "Optimization of mobile games using batching & instancing," IEEE Access, 2021. DOI: 10.1109/ACCESS.2021.3051234

Welden, L. et al., "How retailers and brands thrive in the video game ecosystem," Journal of Retailing, 2025. DOI: 10.1016/j.jretai.2025.03.004