Journal of Applied Multimedia and Networking (JAMN)
Vol.8, No.2, December 2024, pp. 88~94
e-ISSN: 2548-6853

Optimasi Level of Detail dan Occlusion Culling dalam Game Aaron Lost in the Jungle

Frans Anderson Simatupang*, Dwi Amalia Purnamasari**
* Informatics Engineering, Batam State Polytechnic
* * Informatics Engineering, Batam State Polytechnic

Article Info

Article history:

Received Nov 4th, 2024
Revised Dec 5th, 2024
Accepted Dec 23th, 2024

Keyword:

Unity
Level of detail
Occlusion Culling
Game
Optimasi

ABSTRACT

Penelitian ini menganalisis efektivitas teknik optimasi *Culling Occlusion* dan *Level of Detail* (LOD) dalam meningkatkan performa *game* 3D yang tidak optimal. Menggunakan metode penelitian kuantitatif dengan pendekatan eksperimental. *Culling Occlusion* adalah teknik yang menyembunyikan objek yang tidak terlihat oleh pemain, sementara LOD adalah teknik yang menampilkan versi objek dengan detail yang berkurang berdasarkan jarak antara pemain dan objek.Hasil menunjukkan bahwa dengan menggunakan teknik LOD berhasil meningkatkan FPS sebesar 107%,sementara dengan teknik Occlusion Culling hanya meningkatkan secara marginal yaitu 19%.Temuan ini mengindikasikan bahwa meskipun kedua teknik efektif dalam mengoptimalkan performa, teknik LOD memberikan peningkatan yang lebih signifikan.

Corresponding Author:

Dwi Amalia Purnamasari

Informatics Engineering, Batam State Polytechnic
Email: dwiamalia@polibatam.ac.id

1. INTRODUCTION

Video game telah berevolusi menjadi bentuk hiburan yang ada di mana-mana, menawarkan berbagai pengalaman interaktif yang melibatkan pemain di berbagai platform[1].Peminat *Video game* juga semakin meningkat dalam beberap decade terakhir,fenomena ini yang menarik para investor untuk mengembangkan game[2].Pengalaman bermain *game* tradisional sering kali membebani perangkat pengguna karena kompleksitas *game* modern.

Judul-judul yang intensif secara grafis dengan lingkungan 3D yang mendetail dan jumlah efek di dalam permainan membutuhkan daya komputasi yang besar agar dapat berjalan dengan lancar[3]. Pengembangan *video game* menuntut optimasi yang cermat guna menjaga kinerja dan kualitas visual. Teknik mengoptimasi dapat diterapkan disetiap aspek video *game*, seperti pada proses pembuatan *Asset*, kita dapat mengurangi detailnya dengan menggunakan jumlah *polygon* yang rendah[4].Teknik yang digunakan dalam hal ini adalah *Level of Detail* (LOD) dan *Occlusion Culling*[5].

Penelitian ini bertujuan untuk mengoptimalkan game Aaron Lost in The Jungle sehingga dapat dinikmati oleh sebagian besar pengguna. Dengan kinerja dan kualitas visual yang optimal pada berbagai spesifikasi perangkat. Teknik optimasi seperti LOD dan *Occlusion Culling* diharapkan dapat mengoptimalkan kinerja *game* tanpa mengurangi pengalaman bermain.

Studi ini diharapkan dapat memberikan panduan bagi pengembang tentang cara menerapkan teknik optimasi dalam pengembangan *game*. Untuk pemain, hasil pengoptimalan ini diharapkan dapat memberikan pengalaman bermain yang lebih baik, bahkan pada perangkat dengan spesifikasi yang lebih rendah. Selain itu, hasil optimasi ini juga dapat digunakan sebagai referensi bagi pengembang *game* lainnya dalam mengatasi masalah serupa.

e-ISSN: 2548-6853

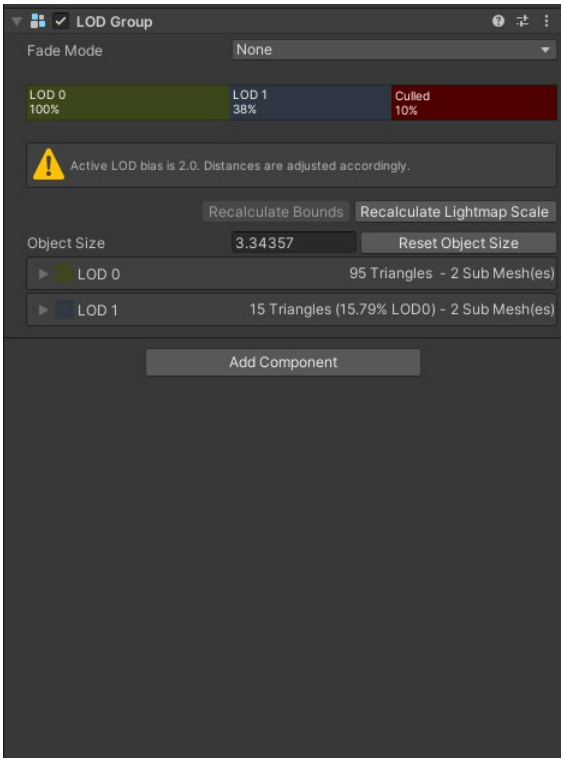
Penelitian ini dibatasi pada *game* Aaron Lost in The Jungle yang diikutsertakan dalam perlombaan KMIPN IV. Fokus penelitian ini adalah pada penggunaan teknik LOD dan *Occlusion Culling* dalam optimasi grafis dan kinerja game. Uji coba dilakukan dengan alat yang memenuhi spesifikasi yang diperlukan untuk menjalankan *game*.

2. RESEARCH METHOD

Metode yang digunakan untuk penelitian ini adalah kuantitatif dengan pendekatan eksperimental. Hal ini melibatkan penciptaan kondisi terkontrol di mana aplikasi yang sama diuji di bawah tiga skenario pengoptimalan yang berbeda: tidak dioptimalkan, dioptimalkan dengan LOD, dan dioptimalkan dengan LOD dan *Occlusion Culling*. Metrik kinerja dikumpulkan dan dianalisis untuk menentukan efektivitas setiap teknik pengoptimalan.

2.1 Level Of Detail

Level of Detail (LOD) dalam grafik komputer mengacu pada teknik yang digunakan untuk mengelola dan mengurangi kerumitan model 3D dalam rendering waktu nyata[6].Dengan menyesuaikan detail model berdasarkan ukuran atau kepentingannya dalam adegan yang dirender, teknik LOD membantu mengoptimalkan kinerja dan mempertahankan ketepatan *visual*. *Game* pada umumnya sering kali melibatkan penyederhanaan *mesh*, di mana simpul, tepi, dan permukaan dikurangi untuk menurunkan kompleksitas geometris[7].Implementasi LOD pada game ini dilakukan dengan menyiapkan 2 model dengan jumlah *triangles* yang lebih rendah dari versi originalnya,yang kemudian ketika jarak pemain dengan objek mencapai titik tertentu maka LOD yang sesuai akan ditampilkan.



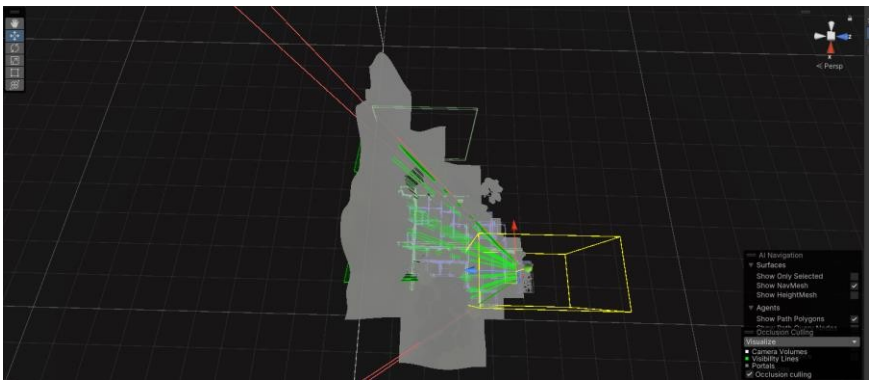
Gambar 1. *Level of Detail* dalam *unity*

2.2 Occulusion Culling

Occlusion culling adalah teknik dalam grafik komputer yang meningkatkan efisiensi rendering dengan tidak menggambar objek yang tersembunyi dari sudut pandang pemain.[8] Metode ini memastikan bahwa hanya permukaan yang terlihat saja yang diproses, secara signifikan mengurangi beban komputasi dan meningkatkan performa *rendering*[9].

Dengan teknik *Occlusion Culling*, *unity* hanya melakukan *rendering* pada objek yang ada dalam *Point Of View*(POV) pemain, sedangkan untuk objek yang tidak di POV pemain dan yang sedang terhalangi oleh objek yang sudah di *render*, tidak akan di *render* oleh *Unity*, Gambar 2 sebagai contoh implementasi dalam *game*.

e-ISSN: 2548-6853



Gambar 2. *Occlusion Culling* dalam *unity*

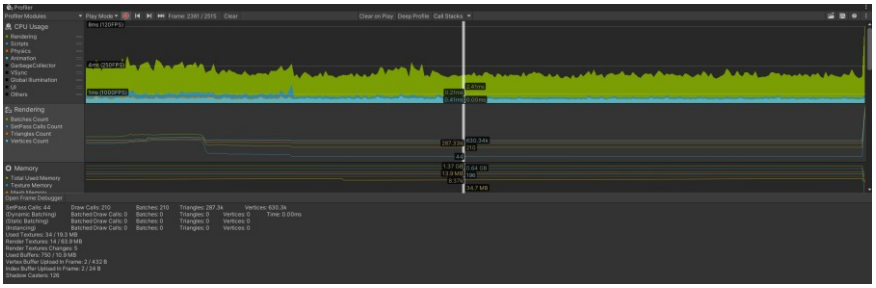
2.3 Unity Profiler

Unity's Profiler adalah alat analisis kinerja tentang perilaku *runtime* aplikasi *Unity*. Dapat diakses di dalam *Unity Editor* atau selama bermain *game*, alat ini menyediakan bagian yang didedikasikan untuk CPU, GPU, memori, dan profil *rendering*. Pengguna dapat mengidentifikasi hambatan kinerja dengan memeriksa penggunaan CPU di seluruh skrip dan fungsi, kinerja GPU termasuk waktu rendering dan efisiensi *shader*, alokasi memori dan pola penggunaan, dan metrik *pipeline* rendering seperti *draw call* dan *batch*[10].

Profiler ini digunakan untuk mengidentifikasikan attribut yang mempengaruhi kinerja *game*,data kemudian akan dibandingkan satu sama lain untuk menentukan persentase perbedaan antara Teknik optimasi dan basis awal *game*,Gambar 3 adalah tipe data yang ditampilkan dari *profiler*.

Beberapa atribut yang akan digunakan dalam perbandingan data antara lain :

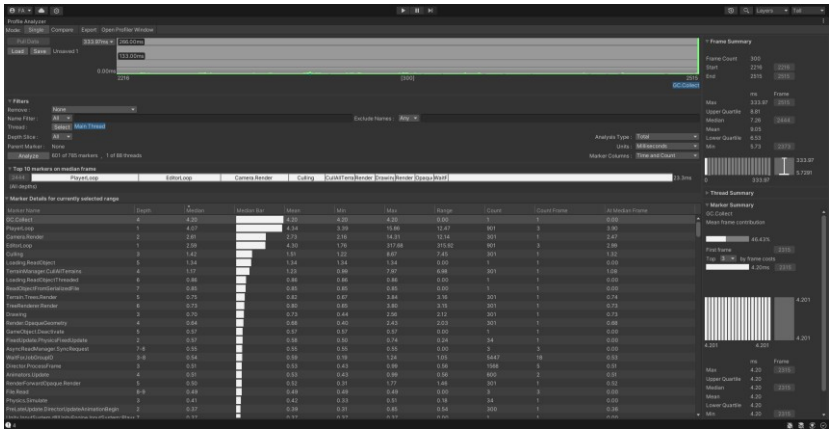
1. *Draw Calls*: Jumlah permintaan kepada GPU untuk merender objek.
2. *Batches*: Berapa banyak draw calls yang berhasil digabungkan menjadi satu batch untuk efisiensi rendering.
3. *SetPass Calls*: Jumlah panggilan yang dibuat untuk mengatur keadaan *render* seperti *material*, *shader*, dan pengaturan lainnya.
4. *Triangles*: Jumlah segitiga yang diterima oleh GPU untuk dirender.
5. *Vertices*: Jumlah titik dalam ruang tiga dimensi yang diterima oleh GPU untuk dirender.
6. *Shadow Casters*: Objek-objek yang melemparkan bayangan di dalam scene.
7. *Used Texture* : jumlah dan jenis tekstur yang digunakan dalam proses *rendering* selama sesi *profiling*. Ini termasuk tekstur-teksur yang diterapkan ke objek
8. *Render Texture*: tekstur yang digunakan sebagai target render untuk operasi-render tertentu dalam aplikasi
9. *Used Buffer* : penggunaan buffer dalam proses rendering. Buffer dalam konteks ini bisa merujuk pada berbagai jenis buffer, termasuk buffer geometri, buffer warna, dan buffer lainnya yang diperlukan untuk menyimpan data dan state selama proses *rendering*.
10. *Vertex Buffer* : tipe *buffer* khusus yang menyimpan data titik-titik dalam ruang tiga dimensi yang akan dirender.



Gambar 3. *Unity Profiler* dalam *unity*

2.4 Unity Profiler Analyzer

Unity Profiler Analyzer adalah alat pendamping *Unity's Profiler*, yang dirancang untuk menganalisis dan menginterpretasikan lebih lanjut data yang dikumpulkan selama sesi *profiling*, fitur-fitur seperti tampilan garis waktu, bagan, dan grafik untuk memvisualisasikan CPU, GPU, memori, dan data *rendering* dari waktu ke waktu dapat kita lihat dengan menggunakan alat ini. Selain itu, terdapat alat untuk membandingkan data *profiling* di antara sesi yang berbeda[11].Fitur yang akan digunakan dalam perbandingan adalah *frame summary* yang berisi tipe data *frame time*,attribut yang akan digunakan adalah *mean* atau rata rata dari kumpulan data,Gambar 4 adalah contoh presentasi data yang dilakukan oleh alat *Unity Profiler Analyer.*



Gambar 4. *Unity Profiler Analyzer* dalam *unity*

3. RESULTS AND ANALYSIS

3.1. Hardware specifications

Tabel 1. Spesifikasi Perangkat

Component	Name
CPU	i5-13500 @ 2.50 GHz (20 CPU)
GPU	Nvidia RTX 3070 8GB
Memory	32 GB

3.2. Metode pengambilan sampel

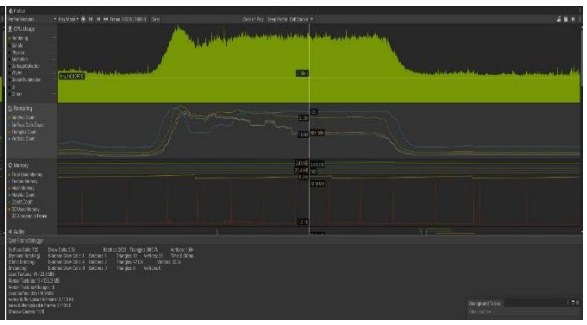
Dalam pengujian permainan ini dilakukan dengan melakukan hal sebagai berikut.pemain akan bergerak dari posisi awal ke kunci pertama,kemudian dari posisi awal ke posisi kunci kedua.kemudian dari posisi kunci kedua ke kunci ketiga,dan terakhir dari posisi kunci ketiga ke kunci keempat,setelah itu akan mengambil nilai tengah dari *Profiler* dengan batas *frame* sebanyak 1000. Dalam pengujian ini. Model yang digunakan sebanyak 3,yaitu model LOD0 yang menggunakan detail 100%,kemudian model LOD1 yang menggunakan detail 50%,dan yang terakhir yaitu model LOD2 yang menggunkan detail 20%,pengurangan detail menggukkan alat decimate yang tersedia di blender.objek yang memiliki 3 model ini objek yang memiliki jumlah yang banyak,seperti pohon,dan batu-batuan.

3.3. Profile (rendering) Graph

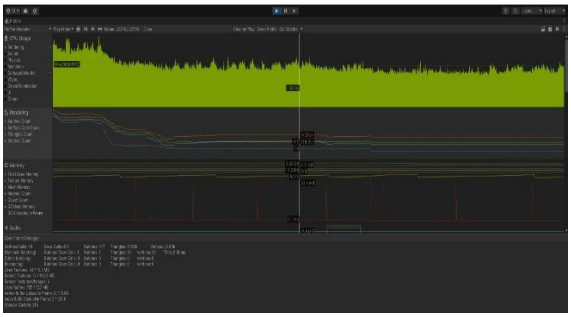
Gambar 5 sampai Gambar 7 merupakan hasil dari *Profiler* yang sudah didapatkan dari ketiga sesi permainan yang telah dilakukan. *Spike* grafik dari masing masing gambar menunjukan adanya penggunaan CPU dan proses rendering yang tinggi ketika pemain mulai memasuki hutan bagian dalam.Perbandingan dari data ini akan disajikan pada Tabel 1.

e-ISSN: 2548-6853

Gambar 5. *Profile* Non-Optimasi



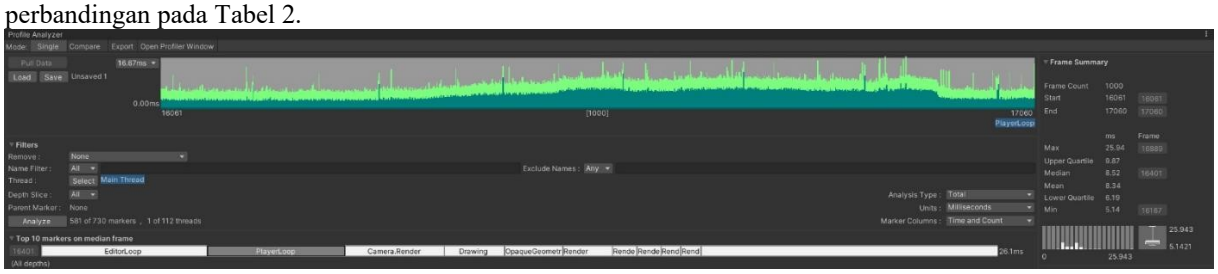
Gambar 6. *Profile Occlusion Culling*



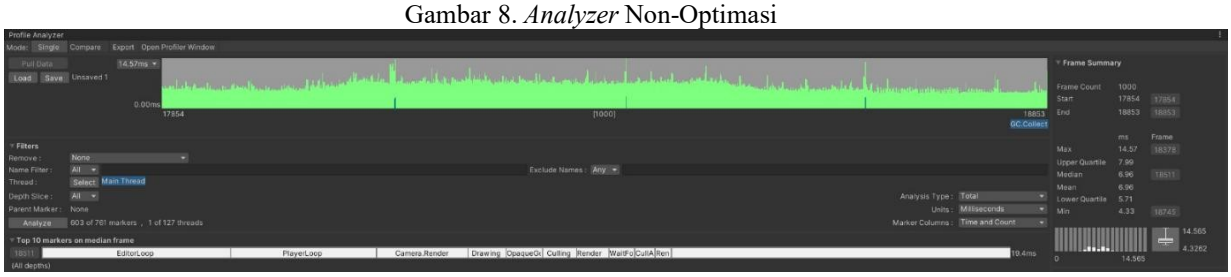
Gambar 7. *Profile Level of Detail*

3.4. Profile Analyzer

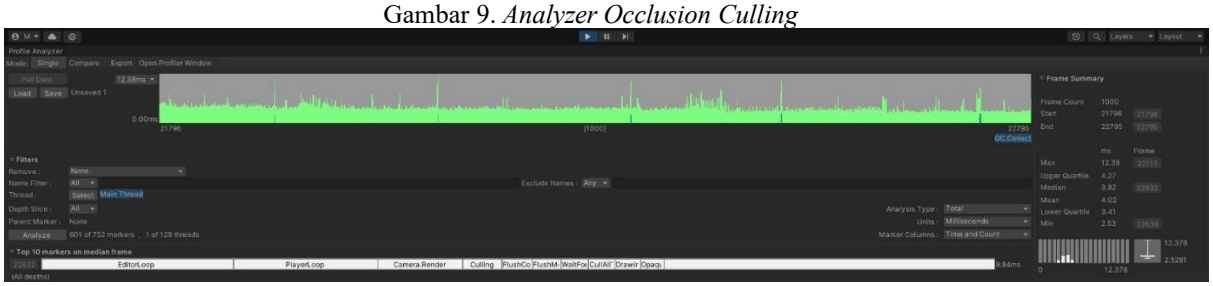
Gambar 8 sampai Gambar 10 merupakan hasil dari *profile rendering* yang diambil dari Gambar 5 sampai Gambar 7, hal yang perlu diperhatikan adalah *Mean* pada kolom *frame summary, spike* grafik menunjukan adanya peningkatan pada bagian tengah sampel yang berarti meningkatnya *frame time* pada bagian itu, setiap gambar memiliki hasil *mean* yang berbeda, hasil ini yang akan di gunakan sebagai data perbandingan pada Tabel 2.



Gambar 8. *Analyzer* Non-Optimasi



Gambar 9. *Analyzer Occlusion Culling*



Gambar 10. *Analyzer Level of Detail*

3.5. Perbandingan Data

Tabel 2. Perbandingan Data Non-Optimasi, Teknik Optimasi *Occlusion Culling* dan Teknik Optimasi *Level of Detail*

Variable	Non- Optimasi	*Occlusion Culling*	*Level of Detail*
Draw Calls	5.5k	2.5K	417
Batches	5521	2502	417
SetPass Calls	1.8k	720	76
Triangles	1.2M	885.7K	172.6K
Vertices	2.1M	1.6M	216.1K
Shadow Casters	1864	1109	215
Used Texture	49/94.7 MB	41/23.8 MB	34 / 18.6MB
Render Texture	13/80.4 MB	13/123.3 MB	13/153.3MB
Used Buffer	682/13.3MB	935 / 16.9MB	755/12.2MB
Vertex Buffer	3/2.7 KB	3 / 1.5KB	3/1,5KB
Mean	8.34	6.96	4,02
FPS	119.904 FPS	143.678 FPS	248.756 FPS

Berdasarkan data dari Tabel 2 diatas,dengan menggunakan rumus ((Non *Optimize* - *Optimize*) / Non *Optimize*) * 100 kemudian di ubah menjadi persentase dengan demikian,Dengan implementasi teknik optimasi *Occlusion Culling*,terjadi peningkat dalam aspek aspek yang dapat mempengaruhi FPS antara lain, *Draw Calls* (-54,55%),*Batches*(-54,68%), *SetPass Calls*(-95,78%),*Triangles*(-85,62%),*Vertices*(-89,71%),*Shadow Casters*(-88,47%),*Used Texture*(-16,33%),*Mean* (-51,80%),*Used Buffer* satu satunya aspek yang mengalami penurunan yaitu dengan (37,10%) sisa 2 aspek tidak mengalami perubahan yaitu *Render Texture* dan *Vertex Buffer,*kemudian untuk metrik performa dari optimasi yaitu FPS,terjadi peningkatan sebanyak (19,83%).

Sedangkan dengan implementasi teknik optimasi *Level of Detail* (LOD) memberikan kontribusi yang sangat jauh dibandingkan teknik *Occlusion Culling* dalam meningkatkan performa antara lain, *Draw Calls* (-92,42%),*Batches*(-92,45%), *SetPass Calls*(-95,78%), *Triangles*(-85,62%), *Vertices*(-89,71%), *Shadow Casters*(-88,47%), *Used Texture*(-30,61%),*Mean* (-51,80%), *Used Buffer* satu satunya aspek yang mengalami penurunan yaitu dengan (10,70%) sisa 2 aspek tidak mengalami perubahan yaitu *Render Texture* dan *Vertex Buffer*, kemudian untuk metrik performa dari optimasi yaitu FPS,terjadi peningkatan sebanyak (107,46%).

4. CONCLUSION

Berdasarkan data yang disajikan, teknik optimasi *Level of Detail* (LOD) telah terbukti lebih efektif dalam meningkatkan performa permainan daripada teknik optimasi Occlusion Culling. Dalam semua elemen utama seperti *Draw Calls*, *Batches*, *SetPass Calls*, *Triangles*, *Vertices*, dan *Shadow Casters*, LOD menimbulkan penurunan yang lebih signifikan daripada *Occlusion Culling*. Hal ini menunjukkan bahwa LOD berhasil mengurangi tugas yang harus dijalankan oleh GPU dan CPU, sehingga menghasilkan peningkatan FPS jauh lebih tinggi daripada *Occlusion Culling*. Di samping itu, LOD juga sukses mengurangi penggunaan tekstur lebih banyak daripada *Occlusion Culling* ,menunjukkan efisiensi dalam pengelolaan sumber daya grafis. Dengan demikian teknik optimasi Level of Detail secara konsisten menghasilkan peningkatan performa yang lebih signifikan dalam permainan dibandingkan dengan teknik optimasi Occlusion Culling, menjadikannya pilihan yang lebih efisien untuk meningkatkan kinerja permainan secara menyeluruh.

ACKNOWLEDGEMENTS

Terima kasih kepada Politeknik Negeri Batam, Digiars Studios, serta tim Xii Studio yang telah membantu dalam hal kerja sama, dukungan serta fasilitas dalam proses penilitian ini.

REFERENCES

[1] S. Egenfeldt-Nielsen, J. H. Smith, and S. P. Tosca, Understanding video games: The essential introduction. 2019. doi: 10.4324/9780429431791.
[2] A. A. Qaffas, "An operational study of video games' genres," International Journal of Interactive Mobile Technologies, vol. 14, no. 15, 2020, doi: 10.3991/IJIM.V14I15.16691.
[3] D. Prayudi and N. Hamdi, "Development of Video Games With WebGL Format That Allows You to Play Video Games Without Need for a Device With High Specifications," Brilliance: Research of Artificial Intelligence, vol. 3, no. 2, 2023, doi: 10.47709/brilliance.v3i2.3004.
[4] G. Koulaxidis and S. Xinogalos, "Improving Mobile Game Performance with Basic Optimization Techniques in Unity," Modelling, vol. 3, no. 2, 2022, doi: 10.3390/modelling3020014.
[5] D. Laksono and T. Aditya, "Utilizing a game engine for interactive 3D topographic data visualization," ISPRS Int J Geoinf, vol. 8, no. 8, 2019, doi: 10.3390/ijgi8080361.
[6] M. M. Keleşoğlu and D. Güleçözer, "A study on digital low poly modeling methods as an abstraction tool in design processes," Civil Engineering and Architecture, vol. 9, no. 7, 2021, doi: 10.13189/cea.2021.091513.
[7] T. Takikawa et al., "Neural Geometric Level of Detail: Real-Time Rendering with Implicit 3D Shapes," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2021. doi: 10.1109/CVPR46437.2021.01120.
[8] L. Ye et al., "3D Model Occlusion Culling Optimization Method Based on WebGPU Computing Pipeline," Computer Systems Science and Engineering, vol. 47, no. 2, 2023, doi: 10.32604/csse.2023.041488.
[9] Y. Li, J. Wang, C. Chen, B. Li, R. Yang, and N. Chen, "Occlusion culling for computer-generated cylindrical holograms based on a horizontal optical-path-limit function," Opt Express, vol. 28, no. 12, 2020, doi: 10.1364/oe.395791.
[10] "Unity - Manual: Profiler overview." Accessed: May 29, 2024. [Online]. Available: https://docs.unity3d.com/Manual/Profiler.html
[11] "Profile Analyzer Window | Package Manager UI website." Accessed: May 29, 2024. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.performance.profile-analyzer@0.4/manual/profiler-analyzer-window.html