Journal of Applied Multimedia and Networking (JAMN)
Vol.x, No.x, Maret 201x, pp. xx~xx
e-ISSN: 2548-6853

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

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

Article Info

Article history:

Received Jun 12th, 201x
Revised Aug 20th, 201x
Accepted Aug 26th, 201x

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. *Occlusion Culling* adalah teknik yang menyembunyikan objek yang tidak terlihat di POV, sementara LOD adalah teknik yang menampilkan versi objek dengan detail yang berkurang berdasarkan jarak antara POV dan objek.Hasil menunjukkan bahwa dengan menggunakan teknik LOD berhasil meningkatkan FPS sebesar 29%,sementara dengan teknik Occlusion Culling tidak dapat meningkatkan performa.Temuan ini mengindikasikan bahwa teknik optimasi LOD memberikan peningkatan yang lebih signifikan dibandingkan teknik optimasi Occlusion Culling.

Corresponding Author:

Third Author,
Departement of Electrical and Computer Engineering,
National Chung Cheng University,
168 University Road, Minhsiung Township, Chiayi County 62102, Taiwan, ROC.
Email: lsntl@ccu.edu.tw

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]. Menurut Steam Charts, sebuah platform yang memonitor aktivitas dan spesifikasi perangkat pengguna, terdapat variasi yang signifikan dalam spesifikasi perangkat keras yang digunakan oleh pemain. Banyak pemain masih menggunakan perangkat dengan spesifikasi menengah hingga rendah, yang sering kali kesulitan menjalankan game modern yang membutuhkan daya komputasi tinggi.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].

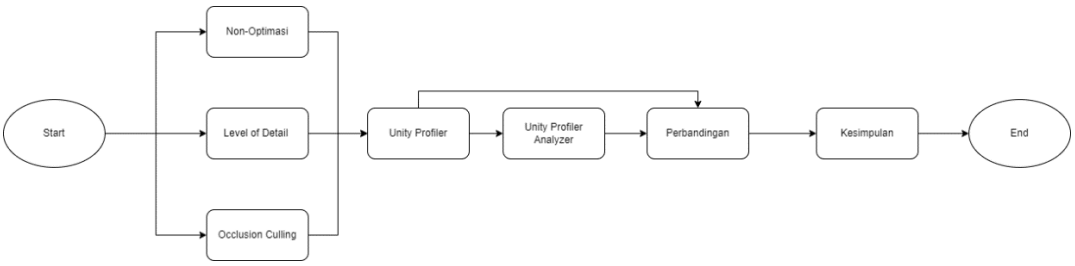
Game Aaron Lost in The Jungle adalah sebuah game petualangan 3D di mana pemain harus mencari kunci untuk membuka kotak ke level selanjutnya. Game ini menampilkan lingkungan yang luas dengan banyak elemen visual seperti vegetasi lebat. Lighting dalam game ini menggunakan teknik pencahayaan statis, mengingat mayoritas dari aset yang digunakan tidak bergerak. Pencahayaan statis diterapkan untuk menghemat sumber daya komputasi. Aset-aset dalam game ini dirancang dalam bentuk low poly untuk mengoptimalkan kinerja, yang mengurangi jumlah polygon tanpa mengorbankan kualitas visual secara signifikan.

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.

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



Gambar 1. Flowchart Metode Penelitian

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

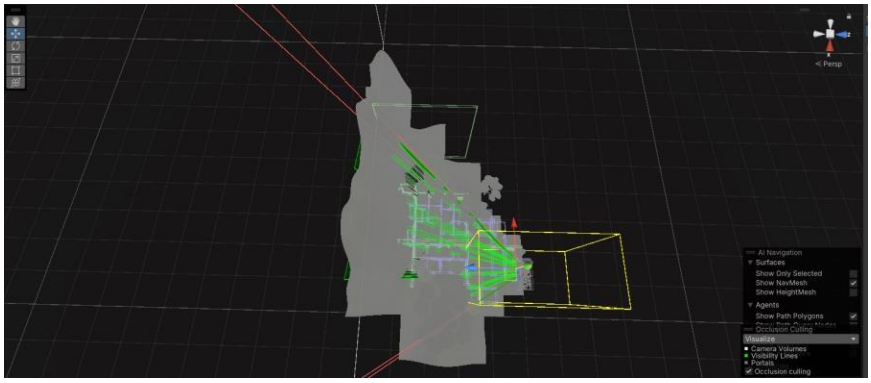
Level of Detail (LOD) dalam grafik komputer mengacu pada teknik yang digunakan untuk mengelola dan mengurangi kerumitan model 3D dalam rendering waktu nyata[6].Dengan menyesuaikan detail model berdasarkan ukuran atau kepentingannya dalam adegan yang dirender, teknik LOD membantu mengoptimalkan kinerja 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 POV dengan objek mencapai titik tertentu maka LOD yang sesuai akan ditampilkan.

Gambar 2. *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 POV.[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), sedangkan untuk objek yang tidak di POV dan yang sedang terhalangi oleh objek yang sudah di *render*, tidak akan di *render* oleh Unity, Gambar 3 sebagai contoh implementasi dalam *game*.



Gambar 3. *Occlusion Culling* dalam Unity

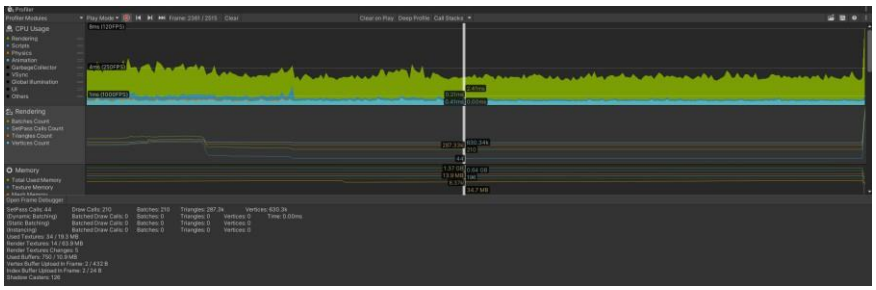
2.3 Unity Profiler

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 4 adalah tipe data yang ditampilkan dari *profiler*.

Beberapa atribut yang akan digunakan dalam perbandingan data antara lain :

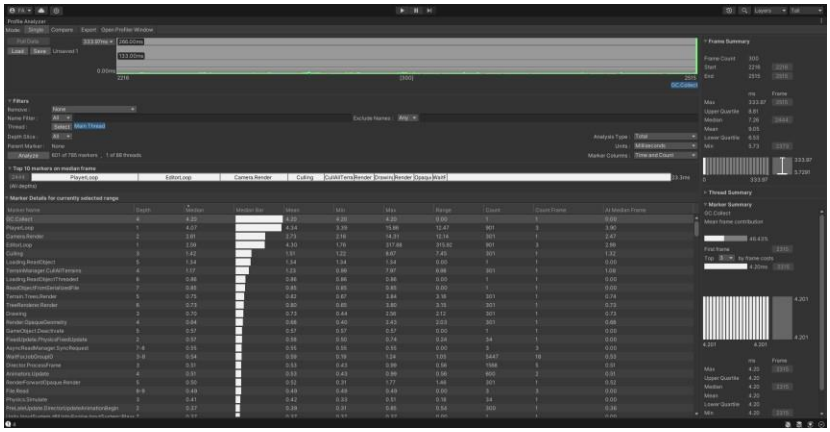
1. *Draw Calls*: Jumlah permintaan kepada GPU untuk merender objek.
2. *Batches*: Berapa banyak draw calls yang berhasil digabungkan menjadi satu batch untuk efisiensi rendering.
3. *SetPass Calls*: Jumlah panggilan yang dibuat untuk mengatur keadaan *render* seperti *material*, *shader*, dan pengaturan 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 4. 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 5 adalah contoh presentasi data yang dilakukan oleh alat Unity *Profiler Analyer*.



Gambar 5. Unity *Profiler Analyzer* dalam Unity

3. RESULTS AND ANALYSIS

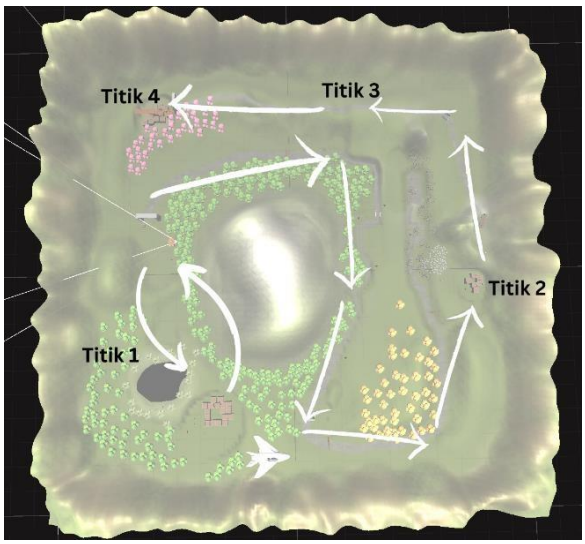
3.1. Hardware specifications

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,Gambar 6 adalah contoh rute yang dilalui sebagai data sampel pada *map*,pengambilan sampel dilakukan 2 kali kemudian dari hasil tersebut diambil rata-rata dari kedua sesi pengambilan data sebagai data perbandingan 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.

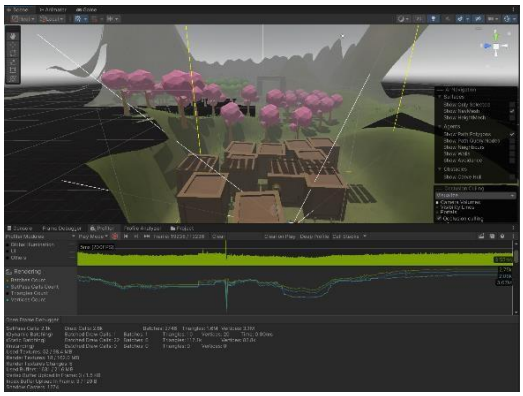


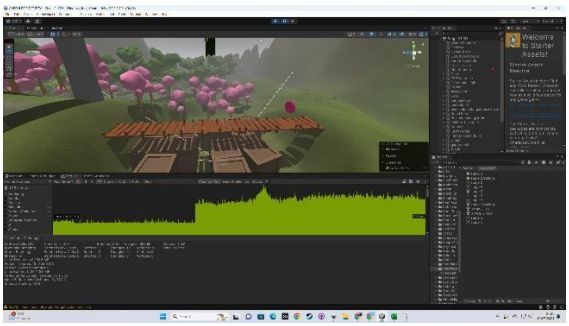
Gambar 6. Rute *gameplay* yang dilalui sebagai sampel

3.3. Profile (rendering) Graph

Gambar 7 sampai Gambar 9 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.

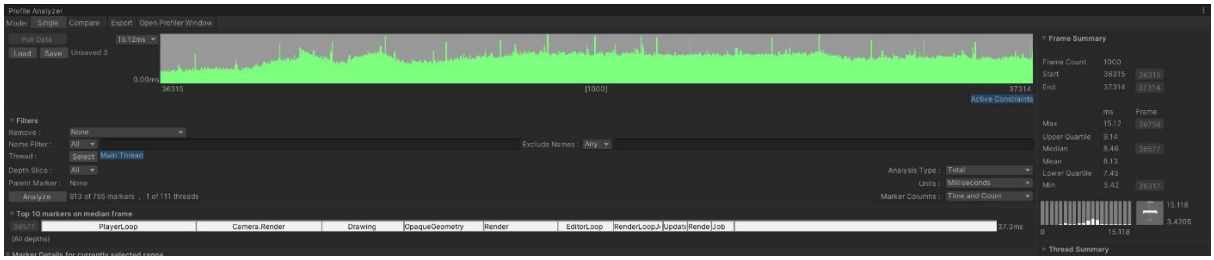

Gambar 7. *Profile* Non-Optimasi


Gambar 8. *Profile Occlusion Culling*

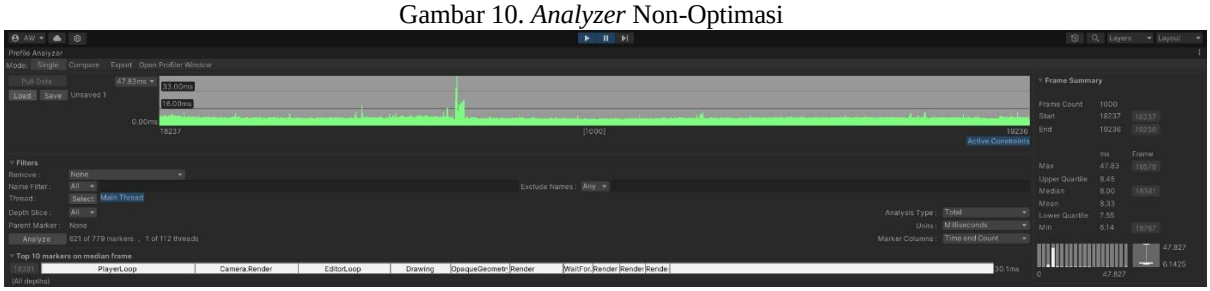

Gambar 9. *Profile Level of Detail*

3.4. Profile Analyzer

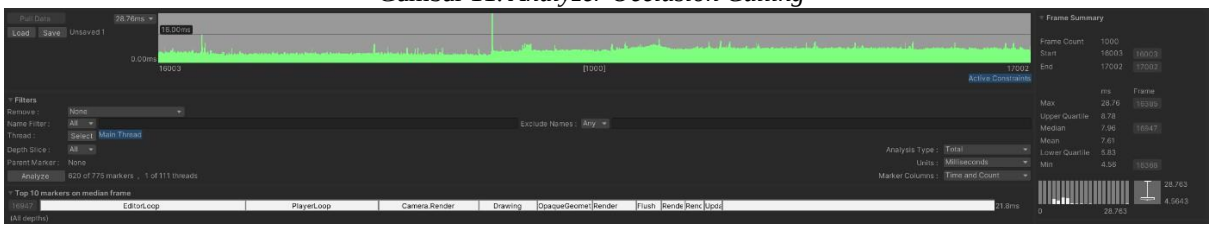
Gambar 10 sampai Gambar 12 merupakan hasil dari *profile rendering* yang diambil dari Gambar 7 sampai Gambar 9, hal yang perlu diperhatikan adalah *Mean* pada kolom *frame summary*, *spike* grafik menunjukan adanya peningkatan aktivitas CPU dan GPU pada sampel yang berarti meningkatnya *frame time* pada bagian itu, setiap gambar memiliki hasil *mean* yang berbeda, hasil ini yang akan di gunakan sebagai data perbandingan pada Tabel 2 dan 3.



Gambar 10. *Analyzer* Non-Optimasi



Gambar 11. *Analyzer Occlusion Culling*



Gambar 12. *Analyzer Level of Detail*

3.5. Perbandingan Data

Tabel 2. Perbandingan data selama 2 sesi gameplay dari Non-Optimasi, Teknik Optimasi *Occlusion Culling* dan Teknik Optimasi *Level of Detail*

Variable	Non-Optimasi 1	*Occlusion Culling* 1	*Level of Detail* 1	Non- Optimasi 2	*Occlusion Culling 2*	*Level of Detail 2*
Draw Calls	3800	2800	3700	3700	2500	3900
Batches	3782	2746	3732	3740	2533	3945
SetPass Calls	3000	2100	213	3000	1900	235
Triangles	1.2M	1.6M	863,6K	1.2M	1.4M	886,8K
Vertices	2.1M	3.1M	1.2M	2.1M	2.7M	1.3M
Shadow Casters	1267	1274	1316	1259	1097	1416
Used Texture	46/92,2MB	52/98,4MB	49/96,5MB	49/96,5MB	51/98,4MB	46/92,2MB
Render Texture	14/143,4MB	18/162,0MB	18/153,8MB	17/158,4MB	17/158,4MB	13/131,8MB
Used Buffer	715/12,0MB	52/98,4MB	1425/15,6MB	761/12,4MB	923/16,8MB	734/12,4MB
Vertex Buffer	3/1,5KB	3/1,5KB	3/1,5KB	3/1,5KB	2/432B	3/1,5KB
Mean	8,13	8,33	7,61	8,07	7,9	4,91
FPS	122.99 FPS	120,05 FPS	131,38 FPS	123,94 FPS	126,58 FPS	203,67 FPS

Tabel 3. Perbandingan data rata-rata Non-Optimasi, Teknik Optimasi *Occlusion Culling* dan Teknik Optimasi *Level of Detail*

Variable	Non-Optimasi	*Occlusion Culling*	*Level of Detail*
Draw Calls	3750	2650	3800
Batches	3761	2639,5	3838,5
SetPass Calls	3000	2000	224
Triangles	1.2M	1.5M	875,2K
Vertices	2.1M	2.9M	1.25M
Shadow Casters	1263	1185,5	1366
Used Texture	47.5/94.35MB	51,5/98,4MB	47,5/94,35MB
Render Texture	15,5/150,9MB	17,5/160,2MB	15,5/142,8MB
Used Buffer	738/12,2MB	487,5/57,6MB	1079,5/14MB
Vertex Buffer	3/1,5KB	2,5/966B	3/1,5KB
Mean	8,1	8,115	6,26
FPS	123,46 FPS	123,21 FPS	159,74 FPS

Berdasarkan data dari Tabel 3 diatas,dengan menggunakan rumus ((Non *Optimize* - *Optimize*) / Non *Optimize*) * 100 kemudian di ubah menjadi persentase dengan demikian,Dengan implementasi teknik optimasi *Occlusion Culling*,terjadi pengurangan dalam beberapa aspek antara lain, *Draw Calls* (-29,33%),*Batches*(-29,81%), *SetPass Calls*(-33,33%), ,*Shadow Casters*(-6,13%),*Used Texture*(-16,33%), *Used Buffer*(-33,94%), *Vertex Buffer*(-16,66%) Sedangkan di beberapa variable ini mengalami jumlah peningkatan dalam jumlah yaitu antara lain *Triangles*(25%),*Vertices*(38,09%), *Mean* (0.18%), *Render Texture* (12.90%) kemudian untuk metrik performa dari optimasi yaitu FPS,terjadi Penurunan sebanyak (0.20%).

Sedangkan dengan implementasi teknik optimasi *Level of Detail* (LOD) dalam meningkatkan performa terjadi pengurangan dalam jumlah antara lain, *SetPass Calls*(-92,53%), *Triangles*(-27,06%), *Vertices*(-40,47%), *Used Texture*(-30,61%),*Mean* (-22,71%), Sedangkan di beberapa variable ini mengalami jumlah peningkatan dalam jumlah yaitu antara lain *Used Buffer* (46,27%) *Draw Calls* (1,33%),*Batches*(2,06%), *Shadow Casters*(8,15%), sisa 3 aspek tidak mengalami perubahan yaitu *Used Texture* ,*Render Texture* dan *Vertex Buffer*, kemudian untuk metrik performa dari optimasi yaitu FPS,terjadi peningkatan sebanyak (29,38%).

4. CONCLUSION

Dari data tersebut, dapat disimpulkan bahwa teknik optimasi Level of Detail (LOD) lebih efektif dalam meningkatkan performa permainan secara keseluruhan dibandingkan dengan teknik optimasi Occlusion Culling. LOD tidak hanya berhasil mengurangi beban kerja GPU dan CPU secara signifikan, tetapi juga meningkatkan FPS yang merupakan indikator penting dalam kinerja permainan. Meskipun ada peningkatan dalam beberapa aspek, peningkatan ini tidak mengurangi efektivitas LOD dalam meningkatkan kinerja permainan. Oleh karena itu, teknik optimasi Level of Detail (LOD) merupakan pilihan yang lebih efisien untuk meningkatkan kinerja permainan secara menyeluruh.

ACKNOWLEDGEMENTS

Terima kasih kepada orang tua yang selalu memberikan dukungan, dan kepada seluruh civitas akademika Politeknik Negeri Batam yang telah memberikan banyak dukungan dan fasilitas selama proses penelitian ini. Terima kasih juga kepada Digiars Studios dan tim Xii Studio atas kerjasama dan dukungan mereka yang sangat berarti dalam penyelesaian penelitian 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