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

Analisis Perbandingan Teknik Optimasi Occlusion Culling & LOD Group Pada Game VR Mahakarya Vokasi Dengan Perangkat VR Oculus Quest 2

Anandika Prima Bayu Kusuma*. Riwinoto*
* Informatics Engineering. Batam State Polytechnic
** Multimedia Engineering Technology Program. Batam State Polytechnic

Article Info

Article history:

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

Keyword:

Virtual Reality
Unity Optimization Graphic
LOD Group
Occlusion Culling
Game

ABSTRACT

Penelitian ini bertujuan untuk mengoptimalkan kinerja grafis dalam pengembangan *game* VR Mahakarya Vokasi dengan menggunakan teknik optimasi *Occlusion Culling* dan *Level of Detail Group* (LOD). Melalui analisis data menggunakan *Profiler* dan *Profiler Analyzer*, penulis menganalisis parameter *rendering* seperti *Setpass Calls, Draw Calls, Batches, Triangles, Vertices, Used Texture, Render Texture, Used Buffers*, dan lain-lain. Hasil penelitian menunjukkan bahwa penggunaan teknik optimasi *Occlusion Culling* dapat signifikan meningkatkan *Frame Per Second* (FPS) sebesar 13,26%, sementara teknik optimasi LOD tidak memberikan perubahan yang signifikan. Meskipun beberapa parameter tidak menunjukkan perubahan yang signifikan, penting untuk terus memantau dan mengoptimalkan parameter ini untuk menjaga kinerja *game* secara optimal. Penelitian ini memberikan panduan yang jelas bagi pengembang *game* VR dengan *gameplay adventure* dalam memilih teknik optimasi yang tepat untuk menciptakan pengalaman pengguna yang lebih stabil dan responsif.

Corresponding Author:

Riwinoto,
Informatics Engineering,
Batam State Polytechnic,
Ahmad Yani St, Teluk Tering, Kec Batam Kota, Batam, Riau Island, 29461, Indonesia
Email: riwi@polibatam.ac.id

1. INTRODUCTION

Dalam pengembangan aplikasi dan *game*, pengoptimalan kinerja grafis adalah salah satu aspek yang sangat penting. Demi mencapai keseimbangan antara visual dan kinerja suatu *game* untuk menciptakan pengalaman pengguna perangkat yang lebih baik. Beberapa strategi yang digunakan untuk mempertahankan tingkat *frame rate* yang tinggi digunakan untuk mencegah rasa pusing pada pergerakan dan memberikan pengalaman visual yang lebih baik dan responsif bagi pengguna. Karena itu beberapa strategi digunakan untuk meningkatkan kinerja pada *game* seperti penggunaan teknik optimasi[1].

Penelitian di bidang optimasi *rendering* untuk *game* terkini memiliki berbagai cabang, diantaranya adalah *Occlusion Culling* dan *Level of Detail Group* (LOD). Terdapat dua pendekatan yang penulis lakukan dalam mengoptimalkan *game* VR Mahakarya Vokasi dengan menggunakan teknik optimasi *Occlusion Culling* dan *Level of Detail Group* (LOD)[2]. *Occlusion Culling* adalah serangkaian proses yang mencegah kamera atau *player* melakukan kerja *rendering* untuk *gameobject 3D* yang benar-benar tersembunyi dari *view* (tertutup) oleh *gameobject 3D* lain pada setiap *frame* kamera menjalankan operasi *culling* yang memeriksa *rendering* dalam *scene* dan mengecualikan (cull) yang tidak perlu ditampilkan[3][4]. *Level of Detail Group* (LOD) merupakan teknik yang memungkinkan penyesuaian detail *gameobject 3D* sesuai dengan jarak pandang kamera atau *player*. Dengan mengaplikasikan *Level of Detail Group* (LOD) objek yang jauh dari pandangan

kamera atau *player* dapat diperkecil dari segi *triangles*, *vertex*, atau tingkat detail lainnya, sekaligus mengurangi jumlah komputasi[4][5].

Temuan dari penelitian sebelumnya mengenai penggunaan teknik optimasi *Level of Detail Group* (LOD) ditemukan lebih efektif dalam mengelola kompleksitas visual dengan mengatur detail objek yang ditampilkan sesuai dengan jarak pandang pemain. Hasil dari penelitian tersebut telah membuktikan bahwa penggunaan *Level of Detail Group* (LOD) mampu meningkatkan kinerja *rendering* secara signifikan. sehingga memberikan pengalaman visual yang lebih mulus dan responsif bagi pengguna[5].

Namun. seiring berkembangnya teknologi dan kebutuhan desain yang berubah. terdapat kebutuhan untuk mengevaluasi ulang metode-metode tersebut dalam konteks pengembangan permainan yang lebih dinamis dan kompleks. Oleh karena itu penelitian saat ini bertujuan untuk mengaplikasikan pembelajaran dari penelitian sebelumnya ke dalam pengembangan *game* Mahakarya Vokasi yang lebih canggih dan menarik.

Penelitian saat ini difokuskan pada pengembangan permainan Mahakarya Vokasi yang menawarkan pengalaman eksplorasi yang lebih mendalam dengan menggunakan 1 *scene* untuk mewakili setiap pulau. Tujuan utama penelitian ini adalah untuk mengevaluasi apakah metode *Level of Detail Group* (LOD) masih layak digunakan dibandingkan dengan metode *Occlusion Culling* dalam konteks pengembangan *game* yang lebih dinamis dan kompleks[5].

Pertanyaan utama yang ingin dijawab pada penelitian kali ini adalah apakah penggunaan metode *Level of Detail Group* (LOD) tetap menjadi pilihan yang optimal untuk mengelola kompleksitas visual dalam lingkungan *game* yang terdiri dari 1 *scene* untuk mewakili setiap pulau, atau apakah metode *Occlusion Culling* dapat memberikan hasil yang lebih baik dalam hal kinerja performa CPU dan GPU, dalam pengalaman pengguna terutama pada visual dan kinerja. Dengan mengeksplorasi perbandingan ini penulis diharapkan dapat memberikan hasil yang berharga bagi pengembang permainan kedepannya dalam memilih metode optimasi grafis yang paling sesuai dengan kebutuhan dan karakteristik dari proyek pengembangan *game* Mahakarya Vokasi.

2. RESEARCH METHOD

Analisis penerapan teknik optimasi pada *game* Mahakarya Vokasi dalam meningkatkan kinerja *frame rate* pada *game* menggunakan metode R & D dengan pendekatan kuantitatif dilihat pada (Figure 1) pengambilan data menggunakan alat pendukung seperti *Profiler*, dan *Profile Analyzer* yang nantinya alat tersebut dapat mengumpulkan data. Dan kemudian membandingkan 3 data yaitu non optimasi, optimasi Occlusion Culling, dan LOD yang nantinya akan dilakukan analisis data untuk dilakukan perbandingan dan kesimpulan penelitian.

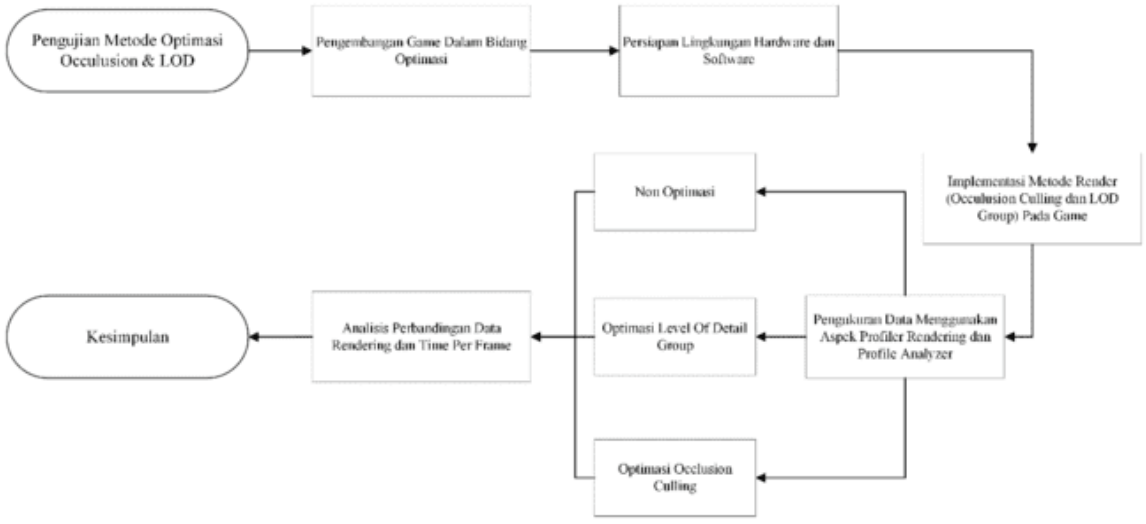


Figure 1.Kerangka Penelitian

2.1. Hardware & Software Specification

Analisis data yang direncanakan akan dilakukan menggunakan dua teknik optimasi *Occlusion Culling* dan *Level of Detail Group* (LOD). Proses analisis akan berfokus pada perangkat keras yang digunakan. termasuk spesifikasi komputer dan perangkat *Virtual Reality* Oculus Quest 2 terlihat dalam (Tabel 1 & 2). Beberapa perangkat lunak pendukung sudah disiapkan untuk kebutuhan analisis data. Ini termasuk *Profiler*, yang akan digunakan untuk mendapatkan nilai dari aspek *rendering* permainan. Selain itu penulis akan menggunakan *Profiler Analyzer* untuk menganalisis data *time per frame* yang merupakan indikator kinerja penting dalam *game* VR. Terakhir penulis akan memanfaatkan *Mesh Optimizer* untuk mengurangi kompleksitas *mesh* pada *gameobject 3D* , sesuai dengan teknik optimasi *Level of Detail Group* (LOD). Analisis akan berfokus pada memahami bagaimana kedua teknik optimasi ini dapat diterapkan dengan efektif dalam konteks perangkat keras dan perangkat lunak yang telah disebutkan. Hal ini akan membantu dalam mengoptimalkan kinerja permainan VR, dan memastikan pengalaman bermain yang mulus dan responsif bagi pengguna Oculus Quest 2.

Tabel 1. Hardware Specification

PC *Spesicfication*	VR *Spesicfication*
Lenovo ThinkStation P340	Oculus Quest 2
Intel Xeon W-1290P CPU @3.70Ghz (20CPUs)	Qualcomm Snapdragon XR2 Octa-core Kryo 585
RAM 32GB	RAM 6GB
VGA Quadro RTX 4000 8GB	Adreno 650
1920 x 1080 Pixel 60Hz	1832 x 1920 Pixel Per Eye 90 Hz
Windows 11	Bluetooth & USB-C Connectivity

Tabel 2. Software Specification

Software	Version
Unity	2021.3.21f1
Profiler Analyzer	1.2.2
Mesh Optimizer	1.1

2.2. Occlusion Culling

Occlusion Culling merupakan teknik yang digunakan dalam pengembangan grafis komputer dengan tujuan mengoptimalkan kinerja *rendering* suatu *game* dan meningkatkan jumlah FPS pada perangkat yang ingin digunakan[6]. Konsep inti dari teknik ini adalah untuk mengurangi jumlah objek yang tidak terlihat atau tidak berada dalam bidang pandang kamera atau *player* dengan cara menghilangkan objek tersebut. Hal ini bertujuan untuk memastikan bahwa hanya objek yang terlihat oleh kamera atau *player* yang di *render*, terdapat beberapa tahapan yang harus diperhatikan sebelum melakukan *Occlusion Culling* (Figure 2)[7][8].

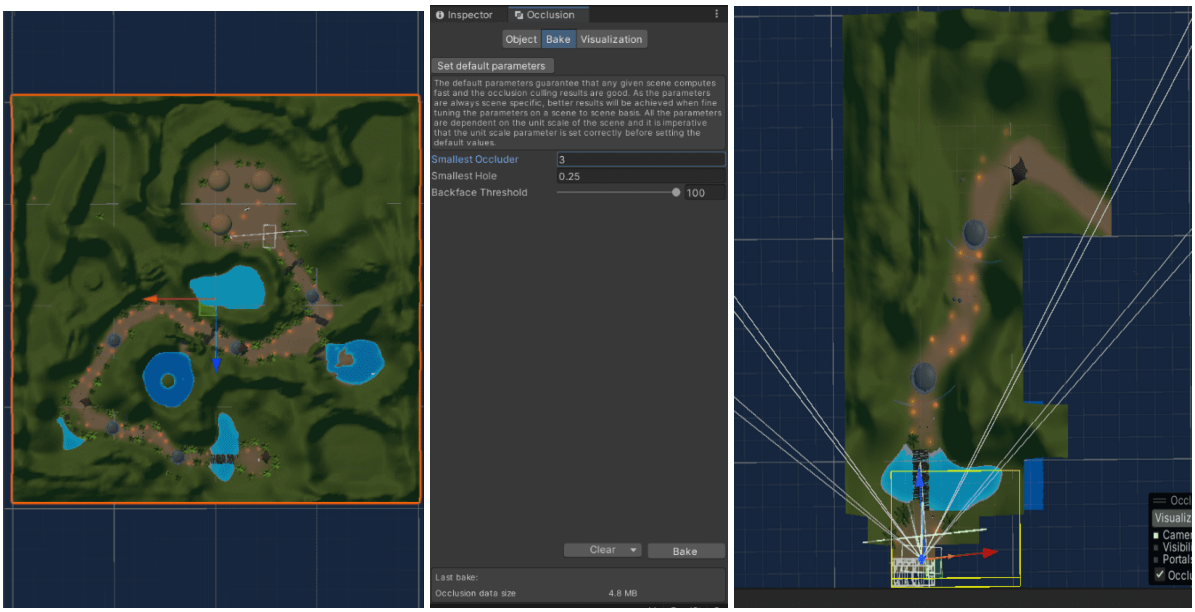


Figure 2. Occlusion Culling

Smallest Occluder adalah salah satu fitur dalam teknik optimasi *Occlusion Culling* dimana *Smallest Occluder* bekerja ketika ukuran *gameobject 3D* terkecil menutupi *gameobject 3D* lain. Secara umum untuk

ukuran *file* terkecil dapat mempercepat *render* dari GPU. Memilih resolusi yang kecil akan memberikan hasil yang maksimal dalam *scene* dan meningkatkan kinerja FPS. Disisi lain menggunakan ukuran *file* yang kecil akan memakan waktu yang lebih lama dalam pemrosesan CPU dan juga menggunakan memori yang lebih banyak ketika *bake* dijalankan. Jika *occluder* terlalu kecil akan ada resiko munculnya *glitch* seperti *gameobject 3D* yang tiba tiba muncul atau menghilang. Hal ini dapat mengganggu pengalaman visual pengguna. Oleh karena itu penting untuk memilih nilai untuk *occluder* dengan baik[8][9].

Smallest Hole adalah fitur dari teknik optimasi *Occlusion Culling* yang menemukan diameter celah terkecil dalam sekumpulan *gameobject 3D* yang menghalangi objek di belakangnya. Seperti yang terlihat oleh kamera untuk *merender gameobject 3D* lain yang mungkin tersembunyi oleh *gameobject 3D* dari kamera dengan menemukan dan menggunakan celah terkecil. Selain itu dengan menguranginya Unity akan lebih mampu membedakan apakah sebuah *occlusion* memang benar-benar *occlusion*, dan proses *bake* dari teknik pengoptimalan *Smallest Hole* akan menghasilkan detail yang lebih tepat. Di sisi lain ini akan memakan waktu lebih lama dan menggunakan lebih banyak memori jika parameter *Smallest Hole* terlalu kecil. Dengan demikian hal ini mengartikan bahwa teknik ini akan memakan waktu lebih lama dan lebih banyak memori di editor Unity[8][9].

Backface Threshold adalah parameter terakhir, yang fungsinya menghilangkan informasi *detail* yang tidak perlu pada permukaan belakang kamera, yang biasanya jarang dilihat oleh kamera. Nilai 100 pada *Backface Threshold* tidak akan pernah menghapus data pada *frame* di belakang, hanya menyembunyikannya. Nilai yang lebih rendah menghasilkan ukuran *file* yang lebih kecil, namun dapat menyebabkan artefak visual. Nilai yang lebih rendah juga akan mengecualikan data *frame* di permukaan belakang dari *Occlusion Culling*. sehingga tidak memerlukan pemrosesan tambahan dan meringankan beban GPU[8][10].

2.3. LOD Group

Level of Detail Group (LOD) digunakan untuk menampilkan versi *gameobject 3D* yang berbeda tergantung pada jarak antara *gameobject 3D* tersebut dengan kamera atau *player*. Ketika *gameobject 3D* menjauh dari kamera atau *player* detailnya menjadi kurang terlihat dikarenakan jumlah *vertex* dan *triangles* kompleks dihapus untuk mengurangi beban pada CPU dan GPU. Dengan menggunakan *gameobject 3D* yang memiliki detail dan *vertex* lebih rendah untuk *gameobject 3D* yang jauh dari pandangan kamera. Penulis dapat mengoptimalkan kinerja tanpa mengorbankan kualitas visual[11].

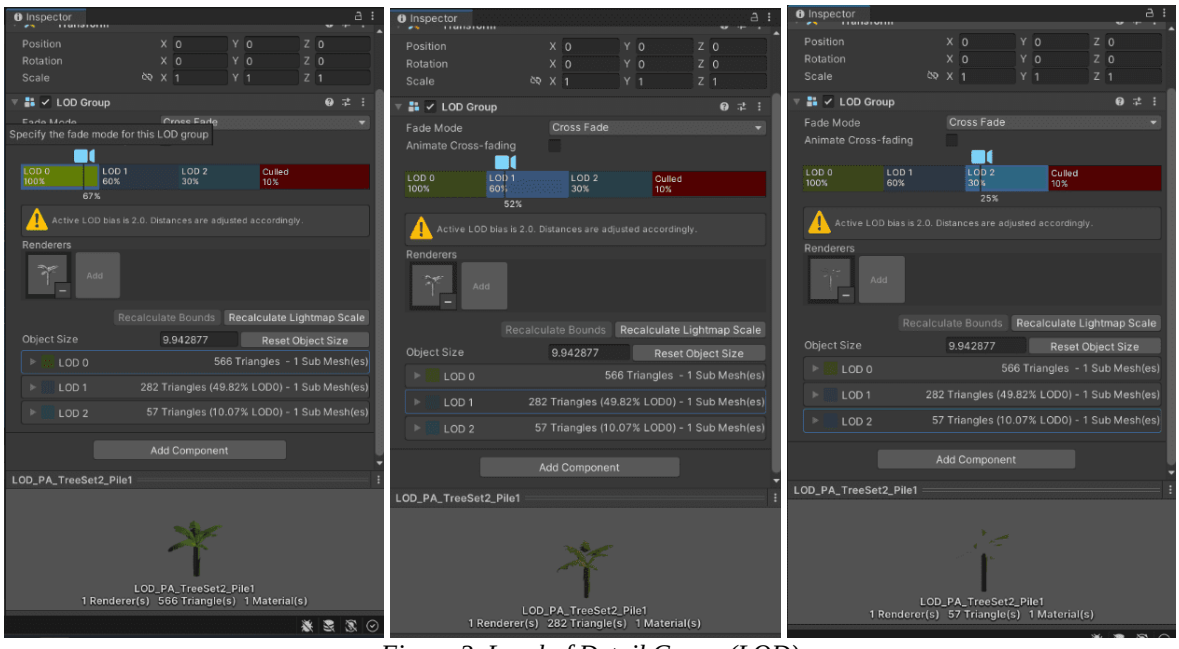


Figure 3. Level of Detail Group (LOD)

Setiap tingkat ini disebut level LOD menandakan semakin jauh *gameobject 3D* dari kamera atau *player* maka semakin rendah tingkat detail dari LOD yang dihasilkan oleh Unity. Teknik ini mengurangi beban pada CPU dan GPU untuk merender *gameobject 3D* yang berada jauh dari kamera atau *player* dan oleh karena

itu dapat meningkatkan kinerja *rendering* sekaligus mengurangi jumlah *vertex* dan *triangles*. Penerapan LOD dapat membantu menciptakan pengalaman visual yang lebih baik dan realistis dalam *game* VR. Dengan mengatur level detail objek-objek berdasarkan jarak kamera atau *player*, LOD dapat membantu mengoptimalkan penggunaan sumber daya komputasi dan mengurangi beban kerja dari GPU[13].

2.4. Aspek *Rendering* dari *Profiler*

Profiler adalah alat yang disediakan oleh Unity untuk menyajikan data kinerja permainan untuk melihat berbagai aspek performa, seperti penggunaan data pada bagian *rendering*, jumlah *Draw Calls*, waktu *render*, dan lain-lain. Dengan informasi ini, penulis dapat mengidentifikasi masalah performa seperti *bottleneck* pada *game*, dan kemudian mengambil tindakan optimasi yang sesuai untuk memperbaiki masalah tersebut[14][15]. Berikut ini adalah aspek yang akan diambil perbandingannya dengan *Profiler:*

- *SetPass Calls*: Jumlah total pemanggilan fungsi *setPass* untuk pemanggilan perubahan material dalam *rendering*. Jumlah setPass yang tinggi dapat mempengaruhi kinerja dari GPU.
- *Draw Calls*: Jumlah total *draw call* yang dikeluarkan Unity dalam satu *frame* permainan. Di saat Unity melakukan *rendering gameobject 3D* ke layar. Ini mencakup panggilan *draw calls* dinamis dan statis.
- *Batches*: Jumlah total *batch* yang dieksekusi dalam satu frame permainan. Jika semakin rendah nilai dari *batches*, hal ini dapat menandakan adanya kemungkinan pemisahan *batch* yang lebih efisien dan meningkatkan kinerja GPU.
- *Triangles*: Jumlah segitiga di *gameobject 3D* yang di *render* dalam satu *frame* permainan.
- *Vertices*: Jumlah total titik *vertex* pada *gameobject 3D* yang di *render* dalam satu *frame* permainan.
- *Used Texture*: Jumlah tekstur gabungan yang digunakan selama satu *frame* permainan dan jumlah memori untuk tekstur yang digunakan.
- *Render Texture*: Jumlah *Render Textures* gabungan di Unity yang digunakan selama satu *frame* permainan dan jumlah memori untuk *Render Textures* yang digunakan.
- *Render Texture Changes*: Jumlah perubahan *render texture* dalam satu *frame*. Jika jumlahnya tinggi. dapat menunjukkan adanya perubahan yang tidak perlu dan mempengaruhi kinerja.
- *Used Buffers*: Jumlah *total buffer* pada GPU dan total memori yang digunakan. Ini termasuk *buffer vertex*, *indeks*, dan komputasi. Serta semua *buffer* internal yang diperlukan untuk *rendering* permainan.
- *Vertex Buffer Upload In Frame*: Jumlah geometri yang diunggah CPU ke GPU dalam satu *frame* permainan. Ini mewakili data *vertex* yang normal atau *texcoord*.
- *Index Buffer Upload In Frame*: Jumlah geometri yang diunggah CPU ke GPU dalam satu *frame* permainan indeks yang menghubungkan *vertex* - *vertex* untuk membentuk segitiga atau poligon.
- Shadow Casters: Jumlah *gameobject 3D* yang menghasilkan bayangan dalam satu *frame* permainan.

2.5. Frame Rate

Dalam dunia *Virtual Reality* (VR) *frame rate headset* (VR) memainkan peran penting dalam memberikan pengalaman yang mulus dan mendalam kepada pengguna. *Frame Rate* diukur dalam satuan *frames per second* (FPS). Menentukan seberapa lancar dan lancar visual di *render* di lingkungan *virtual*[16]. *Frame Rate* yang tinggi secara konsisten dalam *game* (VR) sangat penting untuk menghindari rasa mual atau ketidaknyamanan pada pemain[17]. Dengan itu perlu dilakukannya perhitungan *Frame Rate* agar mengetahui jumlah *frame rate* pada *game* dengan menggunakan perhitungan dibawah ini.

$$Frame\ Rate\ Per\ Second\ (FPS) = \frac{1000}{time\ per\ frame\ (ms)} \quad (1)$$

3. RESULTS AND ANALYSIS

3.1. Prosedur Pengambilan Sampel

Pada (Figure 4) yang menjelaskan dimana penulis mengambil 10 sampel dari salah satu *map* di dalam *game* Mahakarya Vokasi dengan tujuan untuk memperoleh data yang representatif tentang performa *game* pada *map* tersebut. Dengan mengambil 10 sampel dari masing – masing teknik optimasi penulis berharap

mendapatkan gambaran yang lebih akurat tentang pengalaman bermain pada *map* tersebut yang dapat digunakan sebagai dasar untuk mempertimbangkan potensi optimasi atau perbaikan yang diperlukan.

Figure 4. Titik Pengambilan Sampel

3.2. Profiler

Berikut adalah hasil pengambilan 10 sampel dengan bantuan *Profiler* untuk mendapatkan data aspek *rendering* saat permainan dijalankan. Terlihat bahwa nilai dari masing – masing aspek *rendering* menunjukkan peningkatan dan penurunan yang bervariasi tergantung pada jenis optimasi yang digunakan, secara umum data yang dicatat oleh Unity *Profiler* adalah 300 *frame* permainan. Pemantauan *CPU Usage* memberikan data secara *real-time* terkait aspek yang ingin dianalisis sesuai bagian (2.4). Data yang telah dikumpulkan akan ditampilkan pada (Tabel 3,4,5)[14][18].

Table 3.Pengambilan Data Profiler Aspek Rendering

(a) *Non Optimasi*

Variabels	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5	Sampel 6	Sampel 7	Sampel 8	Sampel 9	Sampel 10
Setpass Calls	1300	1500	1100	717	250	576	1200	1000	532	231
Draw Calls	2900	2700	2200	1300	487	1600	2600	2200	1500	571
Batches	2081	2679	2156	1288	486	1605	2590	2199	1459	565
Triangles	5.400.000	4.500.000	3.100.000	1.500.000	328.300	2.100.000	3.600.000	2.400.000	2.100.000	683.700
Vertices	8.100.000	6.800.000	4.400.000	2.100.000	378.700	2.800.000	5.000.000	3.300.000	2.700.000	809.800
Used Texture	58/69.2 MB	49/48.2 MB	49/48.9 MB	51/58.2 MB	45 / 73.5 MB	57 / 65.5 MB	59 / 89.1 MB	49 / 58.6 MB	55/ 64.7 MB	44 / 75.3 MB
Render Texture	32/0.79 GB	36/0.82 GB	36/ 0.80 GB	36/0.82 GB	37 0.80 GB	36 / 0.80 GB	37 / 0.83 GB	36 / 0.80 GB	36 / 0.80 GB	36 / 0.80 GB
Render Texture Changes	21	21	21	20	20	21	21	21	20	20
Used Buffers	2974/115.4 MB	2603/116.6 MB	2588 / 116/4 MB	2365/116.8 MB	2694 / 116.9 MB	2696 / 116.6 MB	2841 / 116.7 MB	2576 / 116.4 MB	2397 / 116.2 MB	2737 / 116.6 MB
Vertex Buffer Upload In Frame	9/2.6 MB	8/1.7 MB	7/1.2 MB	7/470.5 KB	5/ 125.4 KB	7 / 1.1 MB	8 / 1.6 MB	7 / 1.1 MB	9 / 0.9 MB	7 / 365.0 KB
Index Buffer Upload In Frame	9/17.4 KB	8/ 14.1KB	7/13.2 KB	7/8.7 KB	5/4.9 KB	7 /7.3 KB	7 / 14.3 KB	7 / 12.7 KB	7 / 6.8 KB	7/ 5.2 KB
Shadows Caster	1626	1766	1450	743	200	705	1761	1430	683	172

Tabel 4.Pengambilan Data Profiler Aspek Rendering

(b) *Occlusion Culling*

Variabels	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5	Sampel 6	Sampel 7	Sampel 8	Sampel 9	Sampel 10
Setpass Calls	955	1200	1000	717	809	554	712	855	502	277
Draw Calls	2000	2100	2000	1300	1800	1.500	1600	1700	1200	779
Batches	1981	2092	1981	1250	1796	1439	1615	1728	1198	769
Triangles	4.100.000	3.700.000	2.700.000	1.400.000	2.400.000	1.800.000	1.700.000	1.800.000	1.700.000	938.100
Vertices	6.300.000	5.600.000	3.800.000	2.000.000	3.200.000	2.400.000	2.300.000	2.500.000	2.200.000	1.100.000
Used Texture	53 / 63.7 MB	49 / 58.6 MB	48 / 48.6 MB	47 / 47.3 MB	49 / 47.2 MB	42 / 39.9 MB	35 / 36.8 MB	45 / 46.3 MB	46 / 60.2 MB	35 / 39.3 MB
Render Texture	31 / 0.77 GB	31 / 0.77 GB	31 / 0.77 GB	31 / 0.79 GB	32 / 0.80 GB	35 / 0.80 GB	35 / 0.80 GB	36 / 0.83 GB	35 / 0.82 GB	35 / 0.80 GB
Render Texture Changes	21	20	21	21	21	21	20	21	21	21
Used Buffers	7609 / 112.3 MB	7328 / 112.1 MB	7698 / 112.7 MB	7537 / 113.1 MB	8039 / 113.8MB	7713 / 114.4 MB	7562 / 114.1 MB	7642 / 114.3 MB	7724 / 114.2 MB	7687 / 114.4 MB

Variabels	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5	Sampel 6	Sampel 7	Sampel 8	Sampel 9	Sampel 10
Vertex Buffer Upload In Frame	9 / 0.9 MB	7 / 0.9 MB	7 / 0.8 MB	7 / 347.9 KB	7 / 495.0 KB	7 / 0.5 MB	7 / 324.5 KB	6 / 0.8 MB	6 / 0.6 MB	7/ 393.7 KB
Index Buffer Upload In Frame	9 / 10.2 KB	7 / 7.0 KB	7 / 6.7 KB	7 / 4.9 KB	7 / 15.4 KB	7 / 8.1 KB	7 / 6.5 KB	6 / 12.3 KB	6 / 8.5 KB	7 / 5.5 KB
Shadows Caster	1324	1520	1367	660	1080	855	1215	1120	624	248

Tabel 5.Pengambilan Data Profiler Aspek Rendering

(c) *Level of Detail Group* (LOD)

Variabels	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5	Sampel 6	Sampel 7	Sampel 8	Sampel 9	Sampel 10
Setpass Calls	989	1000	793	682	938	949	770	742	469	408
Draw Calls	2300	2400	1800	1600	3200	3300	2400	2300	1.600	1700
Batches	2277	2387	1801	1573	3131	3196	2400	2285	1573	1631
Triangles	3.600.000	2.700.000	2.000.000	1.600.000	3.300.000	3.800.000	2.200.000	2.500.000	1.700.000	2.000.000
Vertices	5.300.000	3.700.000	2.600.000	2.100.000	4.000.000	4.800.000	2.800.000	3.200.000	2.000.000	2.300.000
Used Texture	60 / 90.0 MB	47 / 38.6 MB	47 / 47.3 MB	46 / 38.2 MB	56 / 56.1 MB	57 / 57.4 MB	46 / 47.9 MB	54 / 66.7 MB	52 / 64.7 MB	43 / 37.2 MB
Render Texture	27 / 0.71 GB	27 / 0.59 GB	27 / 0.70 GB	27 / 0.71 GB	27 / 0.71 GB	27 / 0.71 GB	27 / 0.70 GB	27 / 0.70 GB	32 / 0.80 GB	27 / 0.71 GB
Render Texture Changes	21	20	21	21	21	21	21	21	20	21
Used Buffers	3129 / 102.9 MB	2372 / 110.0 MB	2589 / 110.7 MB	2530 / 110.6 MB	2587 / 110.4 MB	2793 / 110.3 MB	2463 / 110.2 MB	4058 / 111.9 MB	4215 / 11.3MB	3823 / 111.6 MB
Vertex Buffer Upload In Frame	9 / 1.5 MB	6 / 1.2 MB	6 / 0.6 MB	6 / 0.6 MB	6 / 2.5 MB	7 / 2.6 MB	7 / 1.0 MB	6 / 0.9 MB	7 / 0.8 MB	7 / 1.6 MB
Index Buffer Upload In Frame	9 / 14.9 KB	6 / 6.4 KB	6 / 5.4 KB	6 / 11.6 KB	6 / 13.7 KB	7 / 13.9 KB	7 / 12.1 KB	6 / 5.3 KB	7 / 5.5 KB	7 / 7.1 KB
Shadows Caster	1468	1557	1207	874	1647	1663	1454	1149	651	586

3.3. Profiler Analyzer

Berdasarkan hasil pengambilan 10 sampel data menggunakan *Profile Analyzer* yang dilengkapi dengan statistik dan visualisasi yang membantu penulis mengumpulkan data *mean* atau *time per frame* untuk setiap 300 *frame*. *Mean* atau *time per frame* adalah waktu yang dibutuhkan CPU dan GPU untuk *merender* satu *frame* permainan. *Profiler Analyzer* menunjukkan *mean* atau *time per frame* yang tinggi dapat mengindikasikan adanya hambatan atau keterlambatan pada CPU dan GPU dalam memproses *frame* tampil ke layar[19]. Data yang telah dikumpulkan akan ditampilkan pada (Tabel 6,7,8).

Tabel 6.Pengambilan Data Profiler Analyzer

(a) *Non Optimasi*

Variabels	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5	Sampel 6	Sampel 7	Sampel 8	Sampel 9	Sampel 10
Mean / Time Per Frame	31.02	40.99	38.18	27.23	17.93	33.37	29.29	28.08	19.75	22.26
FPS	32.24	24.40	26.19	36.72	55.77	29.97	34.14	35.61	50.63	44.92

Tabel 7.Pengambilan Data Profiler Analyzer

(b) *Occlusion Culling*

Variabels	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5	Sampel 6	Sampel 7	Sampel 8	Sampel 9	Sampel 10
Mean / Time Per Frame	27.8	27.12	27.79	27.95	14.83	33.61	23.73	26.49	23.87	21.05
FPS	35.97	36.87	35.98	35.78	67.43	29.75	42.14	37.75	41.89	47.51

Tabel 8.Pengambilan Data Profiler Analyzer

(c) *Level of Detail Group (LOD)*

Variabels	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5	Sampel 6	Sampel 7	Sampel 8	Sampel 9	Sampel 10
Mean / Time Per Frame	28.11	29.15	33.69	31.37	29.62	31.58	28.12	30.71	25.51	20.47
FPS	35.57	34.31	29.68	31.88	33.76	31.67	35.56	32.56	39.20	48.85

3.4. Perbandingan Data Signifikansi Variabel *Profiler* Terhadap Data *Non Optimasi*

Data dari (Tabel 9) menunjukkan hasil rata-rata dari tiga aspek yang berbeda dalam pengambilan 10 sampel, yaitu *Non* optimasi, optimasi *Level of Detail Group* (LOD), dan optimasi *Occlusion Culling*. Pengumpulan data dilakukan dengan bantuan *Profiler* untuk mendapatkan nilai aspek *rendering*, dan *Profiler Analyzer* untuk mendapatkan nilai *mean* atau *time per frame*. Setelah mendapatkan nilai *mean* atau *time per frame*, akan dilakukan perhitungan untuk mendapatkan nilai *Frame Per Second* (FPS) adalah jumlah 1000 / *time per frame* (ms) , seperti yang ditunjukkan pada bagian (2.5).

Tabel 9.Perbandingan Data

Variabel	Non Optimasi	Level Of Detail	Occlusion Culling
Setpass Calls	841	774	758
Draw Calls	1806	2260	1598
Batches	1711	2225	1585
Triangles	2.571.200	2.540.000	2.223.810
Vertices	3.638.850	3.280.000	3.140.000
Used Texture	51 / 64 MB	50 / 54.4 MB	45 / 48.7 MB

Variabel	Non Optimasi	Level Of Detail	Occlusion Culling
Render Texture	35 / 0.79 GB	26.8 / 0.67 GB	33.2 / 0.78 GB
Render Texture Changes	21	21	21
Used Buffers	2.61 / 116.2 MB	2.99 / 99 MB	7.69 / 113.4 MB
Vertex Buffer Upload In Frame	7.4 / 1.4 MB	6.6 / 1.3 MB	7 / 0.5 MB
Index Buffer Upload In Frame	7.1 / 10.4 KB	6.6 / 9.6 KB	7 / 8.5 KB
Shadows Caster	1053	1226	1001
Mean / Time Per Frame	28.81 ms	28.83 ms	25.42 ms
FPS	34.7 FPS	34.6 FPS	39.3 FPS

Kemudian mencari nilai persentase perbandingan *Non* optimasi dengan hasil yang telah dioptimasi menggunakan rumus dibawah ini maka akan didapat nilai persentase pengaruh teknik optimasi tersebut.

$$Persentase = \frac{(Non\ Optimasi - Teknik\ Optimasi)}{Non\ Optimasi}\ x\ 100\% \quad (2)$$

Berdasarkan (Figure 5,6,7) apabila terdapat temuan hasil presentase perbandingan dari nilai yang membandingkan data optimasi antara kondisi *Non* Optimasi, *Level of Detail Group* (LOD), dan *Occlusion Culling*, penulis dapat menyimpulkan bahwa perubahan taraf signifikansi mencapai 5% akan menunjukkan dampak yang cukup berarti atau signifikan pada *Frame Per Second* (FPS), sementara perubahan taraf yang kurang dari 5% dianggap tidak memberikan pengaruh yang signifikan terhadap hasil *Frame Per Second* (FPS)[20].

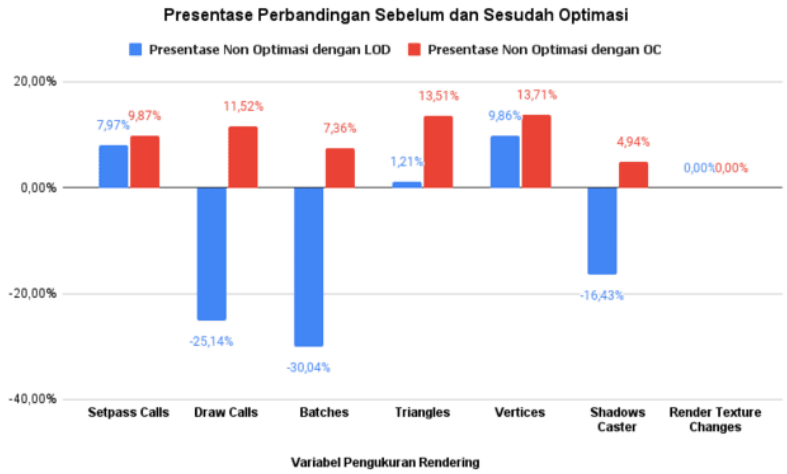


Figure 5. Presentase Perbandingan Data Teknik Optimasi

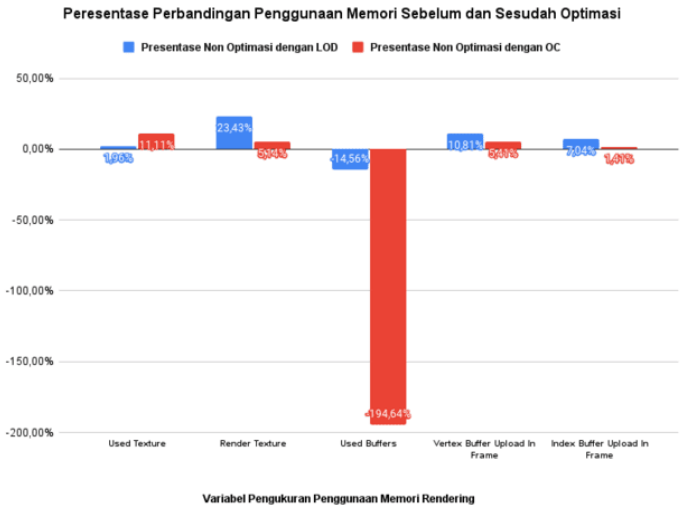


Figure 6. Presentase Perbandingan Penggunaan Memori Teknik Optimasi

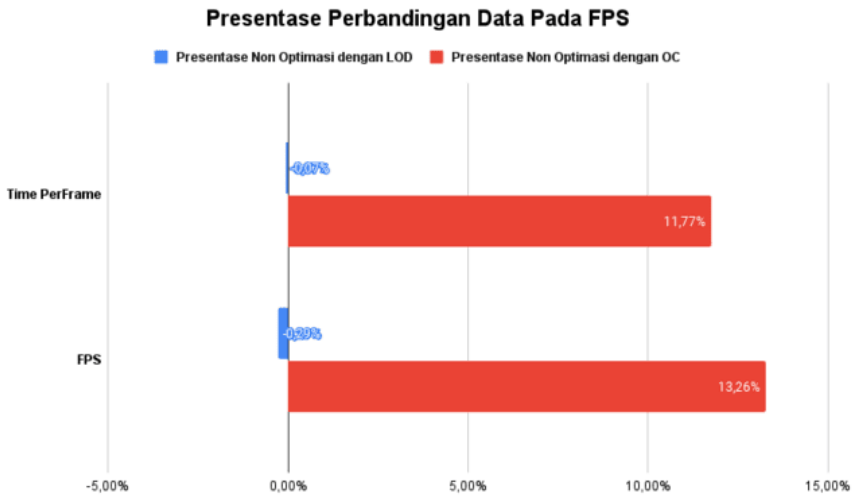


Figure 7. Presentase Perbandingan FPS Teknik Optimasi

Dapat terlihat pada (Figure 5,6,7), beberapa aspek tidak menunjukkan perubahan yang terlalu signifikan setelah dilakukannya teknik optimasi LOD, seperti *Triangles* 1,21%, *Used Texture* 1,96%, dan *Render Texture Changes* yang nilainya tetap sama. Dikarenakan adanya peningkatan signifikan pada *Draw Calls*, *Batches*, *Used Buffer*, dan *Shadow Caster* justru membuat teknik optimasi ini terkesan tidak optimal untuk digunakan, dengan hasil seperti berikut *mean* atau *time per frame* -0.7 dan FPS sebesar -0,29% secara keseluruhan metode optimasi *Level of Detail Group* (LOD) tidak memiliki perubahan yang signifikan, meskipun metode (LOD) berhasil mengurangi beban pada beberapa aspek performa *rendering*, peningkatan pada bagian *Draw Calls*, *Batches*, *Used Buffer*, dan *Shadow Caster* menunjukkan bahwa masih ada ruang untuk perbaikan dalam efisiensi untuk satu *frame* permainan, sehingga kinerja keseluruhan permainan dapat lebih optimal.

Metode *Occlusion Culling* terbukti menjadi pilihan yang sangat efektif dalam meningkatkan kinerja permainan secara signifikan, dengan peningkatan sebesar 39,3 FPS atau sekitar 13,26% dibandingkan dengan kondisi sebelum dioptimasi. Namun dapat dilihat beberapa aspek menunjukkan perubahan yang tidak signifikan. Terdapat, *Shadows Caster* sebesar 4,93%, *Index Buffer Upload In Frame* 1,41% dan *Render Texture Changes* yang nilainya tetap sama. Di sisi lain, terdapat peningkatan penggunaan memory yang sangat signifikan pada *Used Buffers* dengan nilai yang meningkat sebesar 194,64% namun tidak terlalu berpengaruh pada hasil *frame rate per second*. Secara keseluruhan, mayoritas parameter *rendering* menunjukkan penurunan yang signifikan, yang mengindikasikan keberhasilan teknik optimasi *Occlusion Culling* dalam mengoptimalkan penggunaan CPU dan GPU. Hal ini membawa dampak positif yang besar pada pengalaman bermain, dengan memastikan permainan berjalan lebih lancar dan responsif, serta memberikan kinerja yang lebih stabil.

Karena hasil dari teknik optimasi *Level of Detail Group* (LOD) tidak menunjukkan dampak signifikan terhadap *Frame Per Second* (FPS) dengan yang *Non Optimasi*, maka analisis variabel percobaan untuk teknik optimasi *Level of Detail Group* (LOD) tidak dapat dilakukan. Oleh karena itu, penulis hanya akan menganalisis variabel teknik optimasi *Occlusion Culling* yang dibandingkan dengan *Non Optimasi*.

Tabel 9. Signifikansi Teknik Optimasi Terhadap Frame Per Second

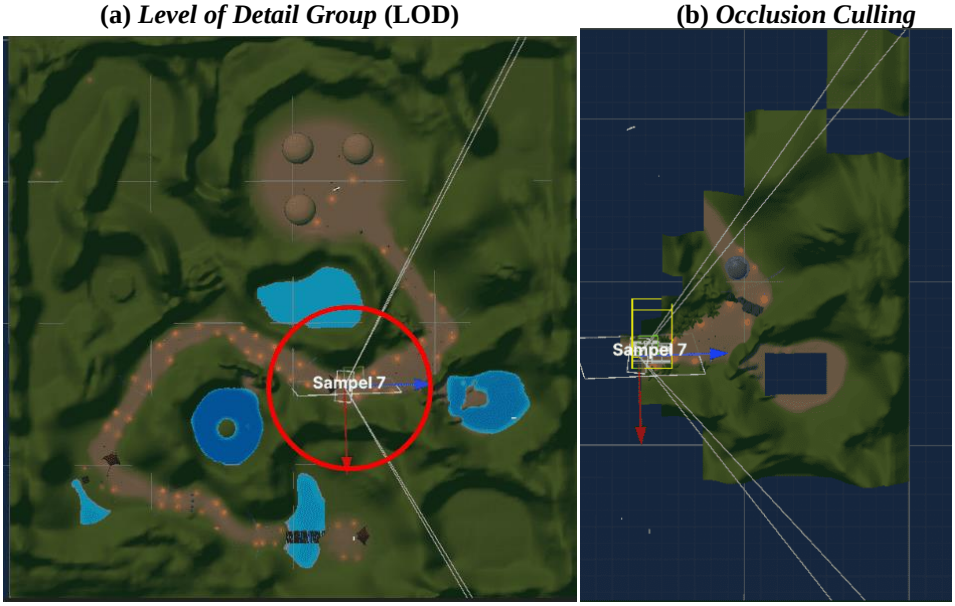
Variabel	*Occlusion Culling* Faktor Signifikan Terhadap Frame Per Second?	*Occlusion Culling* Faktor Signifikan yang Searah atau Tidak Dengan Frame Per Second?	Apakah variabel signifikan terhadap Frame Rate Per Second?
Setpass Calls	YES	NO	YES
Draw Calls	YES	NO	YES
Batches	YES	NO	YES
Triangles	YES	NO	YES
Vertices	YES	NO	YES
Used Texture	YES	NO	YES
Render Texture	YES	NO	YES

Render Texture Changes	NO	-	NO
Used Buffers	YES	YES	YES
Vertex Buffer Upload In Frame	YES	NO	YES
Index Buffer Upload In Frame	NO	-	NO
Shadows Caster	NO	-	NO

Hasil pada (Tabel 9), terdapat temuan signifikansi teknik optimasi terhadap *frame rate per second*, mengacu pada apakah teknik optimasi tersebut meningkatkan variabel secara signifikan atau tidak. Dengan menggunakan asumsi bahwa penurunan atau kenaikan nilai dari variabel *frame per second* lebih dari 5% akan dianggap mempengaruhi performa *frame per second* [20]. Variabel yang mempengaruhi peningkatan kinerja *frame per second* adalah *Setpass Calls*, *Draw Calls*, *Batches*, *Triangles*, *Vertices*, *Used Texture*, *Render Texture*, *Used Buffer*, dan *Vertex Buffer Upload In Frame*. Dari variabel tersebut *Used Buffer* merupakan variabel yang berkebalikan namun tidak mempengaruhi hasil dari *frame rate per second*.

3.5. Analisa Performansi

(a) *Level of Detail Group* (LOD) **(b) *Occlusion Culling***



(c) *Analisis Zona Render*

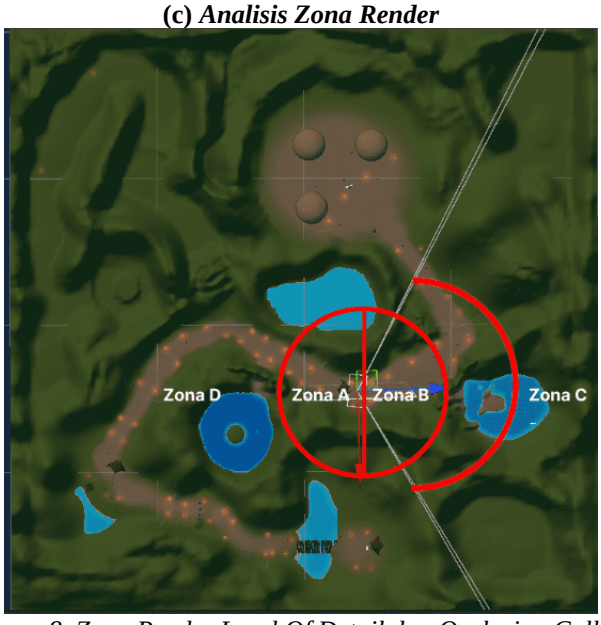


Figure 8. Zona Render Level Of Detail dan Occlusion Culling

Berdasarkan hasil (Figure 8) di atas, posisi kamera atau pemain berada di tengah-tengah peta atau lebih tepatnya pada sampel 7, namun *game* Mahakarya Vokasi adalah *game* di mana pemain akan bergerak secara acak. Hal ini menyebabkan teknik optimasi Level of Detail (LOD) akan merender seluruh peta dari posisi kamera atau pemain, di manapun mereka berada. Asumsi Zona akan digunakan untuk mempermudah, Zona A dan Zona B pada (Figure 7) di atas menjelaskan bahwa LOD akan merender *gameobject 3D* dengan *detail mesh* yang sangat padat atau sering disebut LOD 0, sehingga akan mengecualikan area di luar lingkaran merah atau Zona C dan Zona D yang detail *mesh* kurang padat. Sedangkan *Occlusion Culling* hanya akan *merender* bagian yang terlihat oleh kamera, di mana cakupan *rendernya* adalah Zona B dan Zona C, sehingga dapat mengabaikan Zona A dan Zona D. Dengan penjelasan di atas, dapat diambil kesimpulan bahwa beban *rendering* dan jumlah mesh pada Zona A lebih besar dibandingkan Zona C, sehingga dapat memberikan efisiensi *rendering* pada perangkat dan penurunan signifikan pada beberapa parameter *rendering*. Dengan menggunakan teknik optimasi *Occlusion Culling*, jumlah *mesh* pada *game* Mahakarya Vokasi menjadi rendah sehingga berdampak baik pada performa *game*.

4. CONCLUSION

Berdasarkan hasil penelitian yang telah dilakukan, penelitian kali ini bertentangan dengan penelitian sebelumnya yang dimana teknik optimasi *Level of Detail Group* (LOD) lebih baik dibandingkan dengan *Occlusion Culling*, dan pada penelitian sekarang *Occlusion Culling* lebih baik dibandingkan dengan *Level of Detail Group* (LOD). Penggunaan metode optimasi *Occlusion Culling* berhasil mengurangi jumlah penggunaan di beberapa parameter secara signifikan dengan taraf lebih dari 5% seperti *Setpass Calls, Draw Calls, Batches, Triangles, Vertices, Used Texture, Render Texture, Vertex Buffer Upload In Frame*. Namun, terdapat beberapa parameter yang tidak menunjukkan perubahan signifikan pada perangkat Oculus Quest 2 seperti, *Render Texture Changes, Shadows Caster, Used Buffers* dan *Index Buffer Upload In Frame.* Meskipun beberapa parameter tidak memberikan kontribusi yang signifikan terhadap peningkatan FPS, namun tetap penting untuk memantau dan mengoptimalkan parameter ini untuk memastikan kinerja *game* secara keseluruhan tetap optimal. Hal ini dapat mengindikasikan bahwa penggunaan teknik optimasi *Occlusion Culling* dapat menjadi strategi yang efektif dalam meningkatkan performa *game* dengan mengurangi beban *rendering* dan memori pada CPU dan GPU sehingga dapat menciptakan pengalaman pengguna yang lebih stabil. Namun penelitian lanjutan tetap perlu dilakukan untuk memastikan apakah ada faktor - faktor lain yang menyebabkan teknik *Level of Detail Group* (LOD) kurang optimal dalam meningkatkan *frame per second* dibandingkan dengan *game* yang menggunakan teknik optimasi *Occlusion Culling*, pada kasus *game* yang dimana pemainnya bergerak secara acak mengelilingi peta. Dengan demikian, hasil penelitian diharapkan dapat memberikan panduan yang jelas bagi pengembangan *game* VR Mahakarya Vokasi dalam memilih teknik optimasi yang tepat.

ACKNOWLEDGEMENTS

Terimakasih kepada Politeknik Negeri Batam dan juga ingin menyampaikan terima kasih terkhusus kepada Ibu Kiki Yuliati, selaku Direktur Jenderal Pendidikan Vokasi atas dedikasi dan kontribusinya yang luar biasa dalam mengadakan acara Festival Vokasi x Kampus Merdeka. Tanpa dukungan dan kerjasama yang luar biasa dari semua pihak, acara tersebut tidak akan berhasil seperti yang diharapkan. Terimakasih Kepada Team DigiArs dengan project *game* Mahakarya Vokasi yang membantu penulis dalam proses penelitian.

REFERENCES

[1] M. Tytarenko, “Optimizing Immersion: Analyzing Graphics and Performance Considerations in Unity3D VR Development,” *Asian Journal of Research in Computer Science*, vol. 16, no. 4, pp. 104–114, Oct. 2023, doi: 10.9734/ajrcos/2023/v16i4374.

[2] K. Fathoni *et al.*, “PENGUKURAN PERFORMA ALGORITMA CULLING UNTUK OPTIMASI GRAFIS PADA GAME TIGA DIMENSI CULLING ALGORITHM PERFORMANCE MEASUREMENT FOR GRAPHIC OPTIMIZATION IN THREE DIMENSIONAL GAMES,” vol. 7, no. 1, 2020, doi: 10.25126/jtiik.202071159.

[3] U. Technologies, “Unity - Manual: Occlusion culling.” Accessed: Feb. 20, 2024. [Online]. Available: https://docs.unity3d.com/Manual/OcclusionCulling.html

[4] N. Fedotova, M. Protsenko, I. Baranova, S. Vashchenko, and Y. Dehtiarenko, “RESEARCH ON CALCULATION OPTIMIZATION METHODS USED IN COMPUTER GAMES DEVELOPMENT,” *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska*, vol. 13, no. 3, pp. 37–42, 2023, doi: 10.35784/iapgos.3828.

[5] C. Lisandi Putra, R. Multimedia, and N. Engineering, “Optimization of VR Application Mahakarya Vokasi Land Using Level of Detail (LOD) Technique,” 2023. [Online]. Available: http://jurnal.polibatam.ac.id/index.php/JAMN

[6] U. Engine, “Visibility and Occlusion Culling | Epic Developer Community.” Accessed: Apr. 05, 2024. [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/visibility-and-occlusion-culling-in-unreal-engine?application_version=5.2

[7] N. Fedotova, M. Protsenko, I. Baranova, S. Vashchenko, and Y. Dehtiarenko, “RESEARCH ON CALCULATION OPTIMIZATION METHODS USED IN COMPUTER GAMES DEVELOPMENT,” *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska*, vol. 13, no. 3, pp. 37–42, 2023, doi: 10.35784/iapgos.3828.

[8] U. Technologies, “Unity - Manual: The Occlusion Culling window.” Accessed: May 08, 2024. [Online]. Available: https://docs.unity3d.com/2020.1/Documentation/Manual/occlusion-culling-window.html

[9] R. T. Bonet, “How to Use Occlusion Culling in Unity — The Sneaky Way | TheGamedev.Guru.” Accessed: May 09, 2024. [Online]. Available: https://thegamedev.guru/unity-performance/occlusion-culling-tutorial/

[10] J. Amlin, “Understanding Frustum and Occlusion Culling in Unity3D | by Jared Amlin | Medium.” Accessed: May 09, 2024. [Online]. Available: https://jaredamlin.medium.com/understanding-frustum-and-occlusion-culling-in-unity3d-9b15cc2b078a

[11] A. Edesberg, “Mastering Level of Detail (LOD): Balancing Graphics and Performance in Game Development.” Accessed: Mar. 20, 2024. [Online]. Available: https://www.sloyd.ai/blog/mastering-level-of-detail-lod-balancing-graphics-and-performance-in-game-development

[12] R. T. Bonet, “Level of Detail (LOD) Tutorial for Unity 2021+ | TheGamedev.Guru.” Accessed: May 11, 2024. [Online]. Available: https://thegamedev.guru/unity-gpu-performance/lod-level-of-detail/

[13] U. Technologies, “Unity - Manual: Level of Detail (LOD) for meshes.” Accessed: Mar. 20, 2024. [Online]. Available: https://docs.unity3d.com/Manual/LevelOfDetail.html

[14] U. Technologies, “Unity - Manual: Rendering Profiler module.” Accessed: May 07, 2024. [Online]. Available: https://docs.unity3d.com/Manual/ProfilerRendering.html

[15] U. Technologies, “Unity - Manual: Rendering Profiler module.” Accessed: May 07, 2024. [Online]. Available: https://docs.unity3d.com/2020.1/Documentation/Manual/ProfilerRendering.html

[16] Phil, “What Does The Frame Rate Of A Virtual Reality Headset Indicate - Draw & Code.” Accessed: May 06, 2024. [Online]. Available: https://drawandcode.com/learning-zone/what-does-the-frame-rate-of-a-virtual-reality-headset-indicate/

[17] U. Technologies, “Best practices for profiling game performance | Unity.” Accessed: May 05, 2024. [Online]. Available: https://unity.com/how-to/best-practices-for-profiling-game-performance

[18] U. Technologies, “Unity - Manual: The Profiler window.” Accessed: May 14, 2024. [Online]. Available: https://docs.unity3d.com/Manual/ProfilerWindow.html

[19] U. Technologies, “Optimize Your Game with the Unity Profile Analyzer | Unity.” Accessed: May 14, 2024. [Online]. Available: https://unity.com/how-to/optimize-your-game-unity-profile-analyzer

[20] C. Andrade, “The P value and statistical significance: Misunderstandings, explanations, challenges, and alternatives,” *Indian Journal of Psychological Medicine*, vol. 41, no. 3. Wolters Kluwer Medknow Publications, pp. 210–215, May 01, 2019. doi: 10.4103/IJPSYM.IJPSYM_193_19.