Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

3D Game Asset Optimization Using Quadric Error Metrics And Decimate Modifier

Dani Arifudin*, Nurul Afifatul Khasanah, Ayat Akras Robbani

Computer Science, Information Technology, Universitas Amikom Purwokerto, Banyumas, Indonesia
Email: 1,*daniarif@amikompurwokerto.ac.id, 2nafyy213@gmail.com, 3akrasadlant@gmail.com
Email Penulis Korespondensi: daniarif@amikompurwokerto.ac.id*
Submitted: **18/08/2025**; Accepted: **30/11/2025**; Published: **31/12/2025**

Abstract−3D asset optimization is a crucial challenge in game development because high-polygon models increase computational load, reduce frame rates, and negatively affect user experience in real-time rendering environments. This research addresses the problem by applying the Quadric Error Metrics (QEM) algorithm through the Decimate Modifier (Collapse Mode) in Blender as a solution for mesh simplification. Various high-polygon 3D models were simplified at different reduction levels and tested in the Unity engine to evaluate their impact on performance. The evaluation included frame rate measurement, vertex and triangle count reduction, and geometric accuracy assessment using the Hausdorff Distance method. The results show FPS improvements ranging from 9% to 41%, with vertex reduction between 8% and 33% and triangle reduction between 17% and 22%, while maintaining acceptable visual fidelity. These findings demonstrate that QEM-based optimization effectively balances visual quality and rendering performance, making it suitable for resource-constrained platforms such as mobile games and VR applications.

Keywords: Quadric Error Metrics; Decimate Modifier; Collapse Mode; 3D Optimization; Blender

1. INTRODUCTION

In game development, optimizing 3D assets is essential for maintaining a balance between visual fidelity and rendering performance, especially in real-time environments such as mobile games and virtual reality (VR) applications. High-polygon models increase computational load, reduce frame stability, and affect rendering speed, making efficient optimization indispensable for large and dynamic environments [1]. The increasing complexity of modern game scenes further reinforces the need for techniques that can reliably reduce geometric detail while preserving visual quality [2]. As real-time engines continue to evolve, optimization becomes a central concern that motivates this research [3].

Various mesh simplification techniques have been developed, including Vertex Clustering, Edge Collapse, Surface Clustering, and Quadric Error Metrics (QEM). Vertex Clustering groups vertices within a region; Edge Collapse removes edges by merging vertex pairs; and Surface Clustering simplifies regions based on similar attributes such as normals and color. Among these methods, QEM is widely recognized for its superior ability to preserve geometric characteristics after simplification [4]. QEM calculates a quadric error value to determine optimal vertex positions, allowing the algorithm to maintain sharp features and curvature continuity. It has also benefited from performance improvements enabled by recent parallel processing developments on CPUs and GPUs [1].

Blender provides a practical implementation of QEM through the Decimate Modifier, which includes Un-Subdivide, Planar, and Collapse Mode. Collapse Mode is the most flexible for general-purpose simplification because it applies QEM to reduce polygon count while attempting to retain key geometric features, textures, and object normals [5]. However, simplification may still lead to distortion such as UV stretching, texture artifacts, or loss of normal details, especially on organic or curved objects [6]. To address this, geometric accuracy needs to be evaluated using a robust method. The Hausdorff Distance is commonly used to measure deviation between models before and after simplification and is considered an effective metric for assessing shape changes [7].

In addition to its practical implementation, previous studies emphasize that 3D asset optimization is crucial for balancing visual quality and system performance. High-detail models increase GPU workload, prompting the use of optimization techniques such as Level of Detail (LOD) and mesh simplification to maintain rendering efficiency [8]. Virtanen et al. highlight that mesh decimation techniques are essential for producing lightweight models without sacrificing important details [9]. The QEM approach calculates the quadric error value at each vertex to determine optimal merging operations, offering higher accuracy and lower distortion compared to other simplification methods [3].

Mesh simplification algorithms are commonly categorized into surface clustering, direct sampling, and local geometric element simplification. The local geometric approach, such as Edge Collapse and Triangle Collapse, remains widely used because of its balance between efficiency and accuracy [10]. Vertex Clustering may also be performed using topological, geometric, or semantic criteria [11], while improvements in Edge Collapse methods consider vertex importance or edge characteristics to enhance preservation of shape and structure [12].

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

Blender's Decimate Modifier in Collapse Mode applies Edge Collapse principles based on QEM, enabling the preservation of key features such as textures and normals while reducing polygon count [5]. Polygon reduction targets in game development commonly range from 50% to 70% of the original model, depending on performance needs [13]. To evaluate geometric consistency, the Hausdorff Distance is used to measure the maximum deviation between simplified and original models [14]. Further development may involve integrating LOD systems to load simplified assets adaptively based on camera distance [15]. Previous research also shows that QEM preserves model shape more effectively than clustering-based techniques, especially at higher reduction ratios [16].

Despite extensive research on mesh simplification and the widespread adoption of Quadric Error Metrics (QEM), prior studies primarily focus on algorithmic evaluations rather than practical implementation within real-time game environments. There is still limited empirical evidence on how QEM-based simplification particularly through Blender's Decimate Modifier in Collapse Mode, affects in-engine performance, geometric accuracy, and visual fidelity when deployed in actual gameplay scenarios. The novelty of this study lies in providing a practical, engine-level assessment of QEM simplification by measuring its real impact inside the Unity engine, offering insights not thoroughly explored in earlier works.

Based on these considerations, this research aims to implement and evaluate the QEM algorithm through the Decimate Modifier (Collapse Mode) in Blender as a solution for optimizing 3D models in game development. The objective is to analyze the impact of simplification on performance within the Unity game engine, focusing on frame rate, vertex count, triangle count, and geometric accuracy using the Hausdorff Distance method. Through this study, it is expected that QEM-based optimization can be demonstrated as an effective technique to reduce computational load while maintaining visual fidelity, making it suitable for modern game development, particularly in mobile and VR applications.

2. RESEARCH METHODOLOGY

2.1 Research Design

This research is an experimental study aimed at evaluating the effectiveness of the Quadric Error Metrics (QEM) algorithm in simplifying 3D meshes using the Decimate Modifier (Collapse Mode) in Blender. Blender is chosen in this research due to its open-source flexibility and extensive support for mesh decimation workflows [17]. The experiment was conducted by comparing the results of simplification at various levels of polygon reduction on the visual quality and geometric structure of the object. The evaluation was carried out through comparisons of polygon count, geometric changes, and texture differences using the Hausdorff Distance.

2.2 Research Data and Objects

The dataset used in this study consists of various 3D models with different polygon complexities. The selected models are from asset categories commonly used in games, such as characters, environmental objects, and technical objects.

Model selection criteria:

a. Having a sufficiently high number of polygons before simplification.
b. Having significant detail to test the effectiveness of the QEM algorithm in maintaining visual quality.
c. Having textures that can be used to observe potential distortion due to simplification.

These 3D models were then processed using Blender, with the application of the Decimate Modifier (Collapse Mode) based on the QEM algorithm. As shown in Figure 1, an example of the implementation of the Decimate Modifier in Blender.

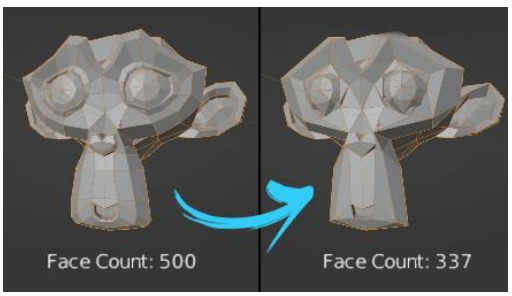

Figure 1. Decimate Modifier

2.3 Research Stages

This research consists of several main stages. The diagram of the research method flow is shown in Figure 2.

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

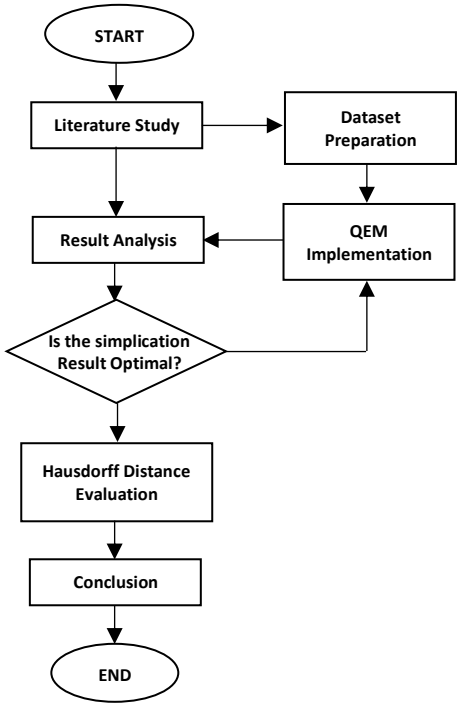

Figure 2. Research Flow

The research methodology is divided into several main stages, as illustrated in Figure 2.

a. Literature Study
This stage aims to collect references and understanding related to the Quadric Error Metrics (QEM) algorithm and 3D mesh optimization methods in game development. The literature used includes scientific journals, books, and technical documentation related to 3D model simplification methods.
b. Dataset Preparation
The dataset used in this study consists of 3D models collected from game asset libraries or created independently. The 3D models are processed into formats compatible with Blender and ready to be simplified using the QEM algorithm.
c. QEM Implementation
The simplification process is carried out using the Decimate Modifier (Collapse Mode) in Blender, which applies the Quadric Error Metrics (QEM) algorithm. Simplification is performed at several levels of polygon reduction, such as 30%, 50%, 70%, and 100%, to observe its impact on model quality. The illustration of this operation can be seen in Figure 3, which shows how an edge is contracted to produce a mesh with fewer polygons without significantly altering the main structure of the model [5].

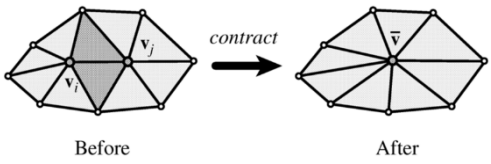

Figure 3. Edge-collapse operation

d. Result Analysis
After simplification is completed, the resulting models are analyzed to assess the effectiveness of the QEM method in reducing mesh complexity. The analysis is carried out by comparing the number of polygons before and after simplification and evaluating geometric changes. The observed reduction in triangle count and vertex data translates directly into fewer draw calls and lighter GPU workloads, improving rendering efficiency [18].
e. Simplification Result Evaluation
At this stage, an evaluation of the simplification results is conducted using the Hausdorff Distance, which measures shape differences between the models before and after simplification. If the simplification results are not optimal,

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

the parameters in the Decimate Modifier can be adjusted, and the QEM implementation process is repeated until the desired results are achieved. As shown in Figure 4, this is the QEM algorithm-simplification effect [3].

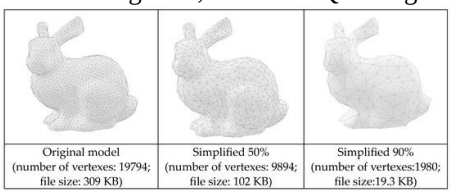

Figure 4. QEM algorithm-simplification effect.

f. Conclusion

The final stage is to summarize the research results based on quantitative and qualitative evaluations. The conclusion is drawn based on the effectiveness of the QEM method in reducing the complexity of 3D models without significantly compromising visual quality.

3. RESULT AND DISCUSSION

3.1 Literature Study

The literature study was conducted by gathering information from various sources such as scientific journals, books, research reports, and online sites relevant to the topics of game development and 3D asset optimization. This step includes researching theories on mesh simplification, game performance, rendering techniques, as well as technical references regarding the QEM algorithm and the Decimate Modifier. In addition, previous studies were analyzed to understand the approaches that have been used in similar contexts. Data obtained from journals, articles, and online statistical platforms were used to strengthen the theoretical foundation and support the applied technical strategies. Additional information was also collected through internet searches to expand the references related to the topic.

3.2 Dataset Preparation

The dataset used in this study consists of 3D assets developed for game simulation purposes. The game assets include environments, common files, characters, NPCs, and various other objects. Some objects were also manually created using Blender 3D software to meet specific needs. Asset sources include the use of free assets from websites such as Sketchfab, Unity Asset Store, Free3D, and CGTrader. Table 1 below shows the character assets used.

Table 1. Character Assets

Image	Item	Description	Source
	Player and NPC	The main character in the game and an NPC serving as a guide.	Model manually created using Blender 3D software
	Environment	The house in this game serves more than just a visual element; it has functional purposes	Model manually created using Blender 3D software
 	Animal	Supporting elements in the game.	Model manually created using Blender 3D software and free asset from https://www.cgtrader.com

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

	Common file	Visual elements within the game	Model manually created using Blender 3D software and free asset from https://sketchfab.com and https://free3d.com

3.3 QEM Implementation

After the creation and collection of 3D assets, the next stage is the implementation of the Quadric Error Metrics (QEM) algorithm using the Decimate Modifier. This implementation aims to determine the extent of vertex and triangle reduction achieved without significantly affecting the visual shape of the object. The implementation was carried out on high-complexity models such as trees, bamboo, plants, and coconut trees. Each object was tested under two conditions: before and after the simplification process. The results can be seen in Figure 5.

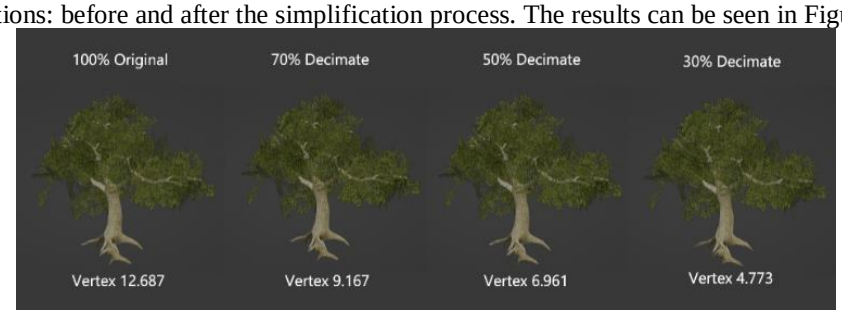

Figure 5. Optimization Result of Tree Model

The following image shows a comparison of the optimized coconut tree model at several levels of decimation. In general, the main structure of the tree remains well preserved despite the reduction in vertex count. The results can be seen in Figure 6.

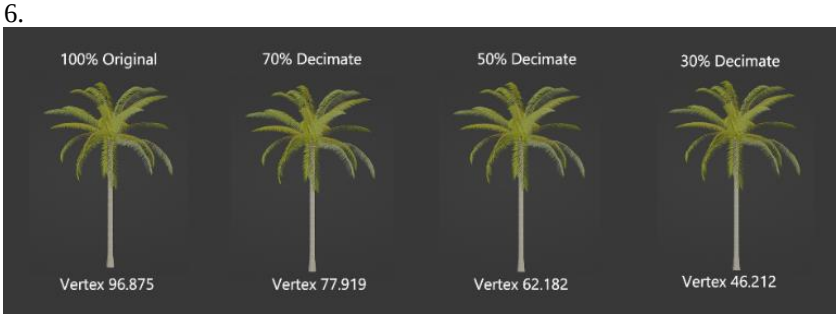

Figure 6. Optimization Result of Coconut Tree Model

The image below presents the optimization results of the bamboo object. The reduction in vertex count was carried out gradually while maintaining the distinctive bamboo shape to ensure visual quality is not compromised within the game engine. The results can be seen in Figure 7.

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

Figure 7. Optimization Result of Bamboo Model

The following visualization shows the optimization results of the plant object at various levels of decimation. The results can be seen in Figure 8.

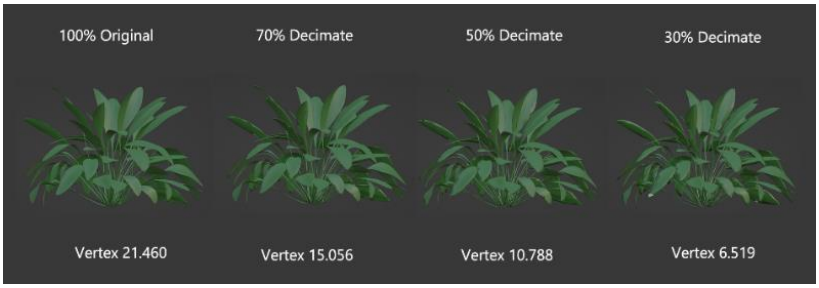

Figure 8. Optimization Result of Plant Model

3.4 Result Analysis

The analysis stage was carried out by implementing the 3D objects—already optimized using the Quadric Error Metrics (QEM) algorithm—into the Unity engine. This was done for the purpose of FPS testing, aimed at evaluating how stable the game performance is under various environmental conditions within the scene. The test was conducted in three different positions within the game, each representing locations with varying levels of visual complexity. The process can be seen in Figure 9.

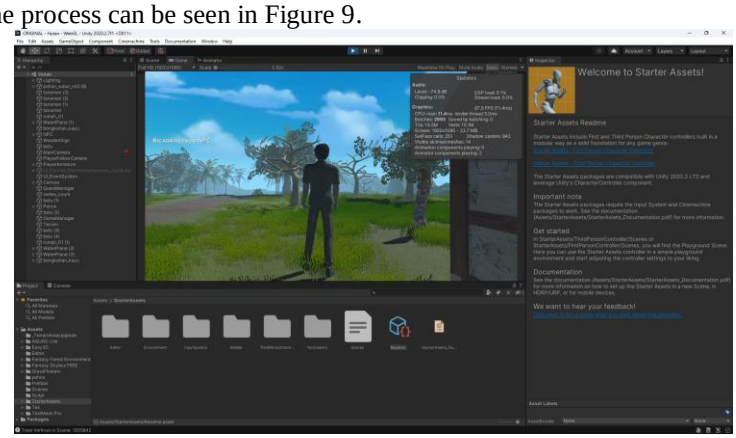

Figure 9. First Position in Forest Scene

The following presents the FPS test results obtained from the first position mentioned earlier. Each position was tested under two different scenarios: without optimization and with optimization using the Quadric Error Metrics (QEM) method and the Decimate Modifier. These results provide insight into how the optimization method affects the stability of game performance under different visual conditions. The results are shown in Table 2.

Table 2. FPS Test Results – First Position in Forest Scene

Object / Scenario	Average FPS	*Vertex Count* at *Render*	*Tris Count* at *Render*	Description
Without QEM Decimate Modifier	80	15,1 Million	13,5 Million	No optimization applied

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

With QEM Decimate Modifier	113	12,6 Million	10,5 Million	Model simplified using the QEM algorithm.

The testing was then continued at the second position, where the player character is located in the middle of the forest area. This can be seen in Figure 10..

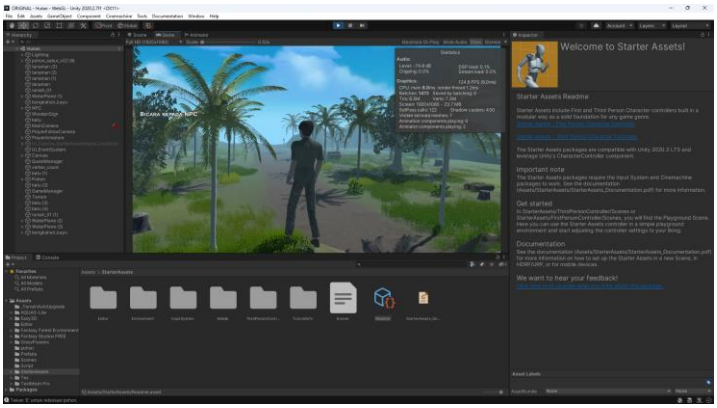

Figure 10. Second Position in Forest Scene

The table presenting the results of the test conducted at the second position in the forest scene is shown in Table 3

Table 3. FPS Test Results – Second Position in Forest Scene

Object / Scenario	Average FPS	*Vertex Count* at *Render*	*Tris Count* at *Render*	Description
Without QEM Decimate Modifier	110	7,3 Million	6,3 Million	No optimization applied
With QEM Decimate Modifier	120	6,7 Million	5,2 Million	Model simplified using the QEM algorithm.

The testing then proceeded to the third position, where the player character is at the edge of the terrain similar to the first position where the character initially appears. This can be seen in Figure 11.

Figure 11. Third Position in Forest Scene

The table presenting the results of the test conducted at the third position in the forest scene is shown in Table 4.

Table 4. FPS Test Results – Third Position in Forest Scene

Object / Scenario	Average FPS	*Vertex Count* at *Render*	*Tris Count* at *Render*	Description
Without QEM Decimate Modifier	53	23,6 Million	21,2 Million	No optimization applied
With QEM Decimate Modifier	65	20,3 Million	16,7 Million	Model simplified using the QEM algorithm.

Subsequently, testing was conducted on the river scene. Although this area is smaller, the number of vertices before rendering reached 3.2 million, compared to around 2 million in the forest area. These figures represent the original

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

(non-optimized) versions, before applying the Quadric Error Metrics (QEM). For the river scene, testing was conducted in only one position due to the limited area size. The first position is shown in Figure 12.

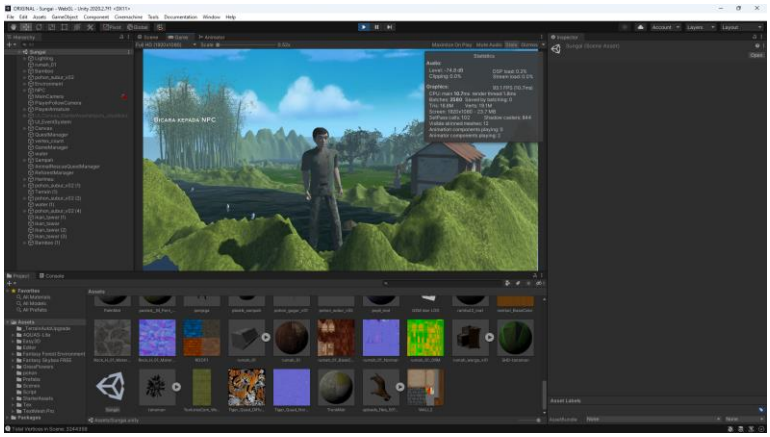

Figure 12. First Position in River Scene

The table presenting the results of the test conducted at the first position in the river scene is shown in Table 5.

Table 5. FPS Test Results – First Position in River Scene

Object / Scenario	Average FPS	*Vertex Count* at *Render*	*Tris Count* at *Render*	Description
Without QEM Decimate Modifier	80	19,1 Million	16,8 Million	No optimization applied
With QEM Decimate Modifier	90	12,8 Million	13 Million	Model simplified using the QEM algorithm.

3.5 Simplification Result Evaluation

The following image shows the version of the model used in the test. The comparison results indicate that, although the QEM algorithm was applied for optimization, there were no significant visual changes. By successfully reducing computational load, the game's visual quality remains preserved. This is due to the fact that the object's basic shape, textures, and visual details remained consistent—especially at close viewing distances. The results can be seen in Figure 13.

Figure 13. Original (Non-Optimized) Model Visualization

The original visual appearance of the model without optimization is shown. This version retains all geometric details of the object without any reduction in vertices or mesh simplification, providing the highest visual quality but with significantly higher computational load compared to the optimized version.
To reduce model complexity without significantly compromising visual quality, the QEM algorithm was applied. The visual appearance of the model after optimization using QEM is shown. The results can be seen in Figure 14.

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

Figure 14. QEM-Optimized Model Visualization

The analysis stage was carried out by implementing the 3D objects optimized using the Quadric Error Metrics (QEM) algorithm via the Decimate Modifier into the Unity Engine. The testing focused on game performance in terms of average frames per second (FPS), by comparing conditions before and after optimization. Each test was conducted in three different positions within the forest scene and one position in the river area, each featuring varying levels of visual complexity. Table 6 summarizes the test results across the different locations.

Table 6. Summary of Result Analysis

Location	FPS Without Optimization	FPS After Optimization	Improvement (%)	Vertex Reduction	Tris Reduction
Position 1 (Forest)	80	113	+41,25%	15,1M → 12,6M (-16,56%)	13,5M → 10,5M (-22,22%)
Position 2 (Forest Center)	110	120	+9,09%	7,3M → 6,7M (-8,21%)	6,3M → 5,2M (-17,46%)
Position 3 (Forest Edge)	53	65	+22,64%	23,6M → 20,3M (-13,98%)	21,2M → 16,7M (-21,23%)
River Position	80	90	+12,5%	19,1M → 12,8M (-32,98%)	16,8M → 13M (-22,61%)

The performance analysis demonstrated that QEM-based simplification significantly improved rendering efficiency across various scenes while maintaining acceptable visual quality. However, performance metrics alone do not fully represent the impact of mesh simplification. Therefore, an additional geometric evaluation was conducted using the Hausdorff Distance to assess the degree of shape preservation between the original and simplified models.

The Hausdorff Distance provides a quantitative measurement of the maximum and average deviation between two meshes. This metric is particularly suitable for evaluating changes that occur due to polygon reduction, as it captures both local and global geometric distortions. Lower values indicate that the simplified model closely retains the original surface characteristics.

Table 7. Hausdorff distance evaluation for simplified 3D models

Model	Reduction	HD Max	HD Mean	HD RMS	Assessment
Tree	60%	0.0082	0.0019	0.0035	Low Deviation
Coconut Tree	55%	0.0067	0.0014	0.0028	Low Deviation
Bamboo	50%	0.0041	0.0009	0.0017	Low Deviation
Plant	45%	0.0034	0.0007	0.0013	Very Low Deviation
Environment Obj.	70%	0.0095	0.0023	0.0041	Moderate but Acceptable Deviation

The Hausdorff Distance evaluation in Table 7 illustrates the geometric fidelity of each model after simplification. The Tree model produced an HD Max value of 0.0082, reflecting minor local distortions in regions with dense branching and high polygon detail. The Coconut Tree model showed slightly lower deviation with an HD Max of 0.0067, indicating that QEM-based simplification was able to retain the overall structure of moderately complex foliage. In both cases, the HD Mean and HD RMS values remained low—ranging from 0.0014 to 0.0035—confirming that the deviations are not widespread across the surface.

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 7, Nomor 2, Desember 2025
e-ISSN 2685-998X
DOI 10.30865/json.v7i2.8974

Simpler models demonstrated even better preservation. The Bamboo model recorded an HD Max of 0.0041, while the Plant model showed the lowest deviation in the dataset with an HD Max of 0.0034 and an HD Mean of 0.0007. These results indicate that the QEM algorithm performs exceptionally well on linear or smooth-surface geometry, where vertex merging introduces minimal visible distortion.

The Environment Object, which presents more irregular geometry and underwent the highest reduction level (70%), produced the largest HD Max value at 0.0095. Despite this, the HD Mean (0.0023) and HD RMS (0.0041) values remain within acceptable thresholds for real-time rendering. The deviation is categorized as moderate but still visually acceptable, indicating that even aggressive simplification can maintain structural fidelity for large environmental assets.

Overall, the numerical trends in Table 7 confirm that Blender's Decimate Modifier (Collapse Mode), powered by the QEM algorithm, maintains geometric accuracy across a range of model complexities. When combined with the FPS improvements reported earlier, these findings demonstrate that QEM-based simplification provides an optimal balance between polygon reduction and visual fidelity, making it highly suitable for performance-constrained real-time applications in the Unity engine.

4. CONCLUSION

This study demonstrates that the implementation of the Quadric Error Metrics (QEM) algorithm through Blender's Decimate Modifier (Collapse Mode) is effective for optimizing 3D assets in real-time game development. Performance testing across multiple Unity in-game scenes showed FPS improvements ranging from 9% to 41%, accompanied by reductions in vertex counts between 8% and 33% and triangle counts between 17% and 22%. Despite these reductions, the visual fidelity of the models remained acceptable, indicating that QEM can significantly decrease geometry complexity without degrading perceptual quality. The Hausdorff Distance evaluation further confirmed that geometric deviations were minimal, with HD Max values ranging from 0.0034 to 0.0095 depending on model complexity. These results indicate that QEM-based simplification preserves essential structural details while enabling substantial performance gains, particularly in high-polygon environments. The novelty of this work lies in its practical, engine-level evaluation of QEM simplification within the Unity game engine, combining both performance metrics and geometric accuracy assessment—an aspect that has received limited attention in previous studies. This integrated evaluation provides a more comprehensive understanding of how mesh simplification affects real-time rendering performance. Overall, the findings highlight QEM as a reliable and efficient method for optimizing 3D assets in resource-constrained platforms such as mobile and VR. Future research may explore adaptive Level of Detail (LOD) integration, automated optimization pipelines, or comparisons with emerging machine-learning-based mesh simplification methods to further enhance asset performance in modern game engines.

ACKNOWLEDGMENT

The authors would like to express their deepest gratitude to the Institute for Research and Community Service (LPPM) of Universitas Amikom Purwokerto for the support provided in the implementation of this research. The assistance and facilitation offered by LPPM greatly contributed to the smooth progress of the research, from the planning stage, implementation, to the preparation of the final report. This support played a significant role in achieving the objectives of this study, particularly in the development and optimization of 3D graphic technology for educational purposes and the game industry.

REFERENCES

[1] Q. Zhang, "Advanced techniques and high-performance computing optimization for real-time rendering," *Appl. Comput. Eng.*, vol. 90, no. 1, pp. 14–19, 2024, doi: 10.54254/2755-2721/90/2024melb0061.

[2] Y. Ding and Y. Song, "Vision-Degree-Driven Loading Strategy for Real-Time Large-Scale Scene Rendering," *Computers*, vol. 14, no. 7, 2025, doi: 10.3390/computers14070260.

[3] H. Chang, Y. Dong, D. Zhang, X. Su, Y. Yang, and I. Lee, "Review of Three-Dimensional Model Simplification Algorithms Based on Quadric Error Metrics and Bibliometric Analysis by Knowledge Map," *Mathematics*, vol. 11, no. 23, 2023, doi: 10.3390/math11234815.

[4] T. Zhao, L. Busé, D. Cohen-Steiner, T. Boubekeur, J. M. Thiery, and P. Alliez, *Variational Shape Reconstruction via Quadric Error Metrics*, vol. 1, no. 1. Association for Computing Machinery, 2023.

[5] Blender Foundation, "Decimate Modifier - Blender Manual," 2025.

https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/decimate.html.

[6] L. Szirmay-Kalos and M. Magdics, "Adapting Game Engines to Curved Spaces," *Vis. Comput.*, vol. 38, no. 12, pp. 4383–4395, 2022, doi: 10.1007/s00371-021-02303-2.

[7] Z. S. Gharehtappeh and Q. Peng, "Simplification and unfolding of 3D mesh models: Review and evaluation of existing tools," *Procedia CIRP*, vol. 100, no. March, pp. 121–126, 2021, doi: 10.1016/j.procir.2021.05.023.

[8] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," *Proc. 24th Annu. Conf. Comput. Graph. Interact. Tech. SIGGRAPH 1997*, pp. 209–216, 1997, doi: 10.1145/258734.258849.

[9] J. P. Virtanen *et al.*, "INTERACTIVE GEO-INFORMATION in VIRTUAL REALITY - OBSERVATIONS and FUTURE CHALLENGES," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. - ISPRS Arch.*, vol. 44, no. 4/W1, pp. 159–165, 2020, doi: 10.5194/isprs-archives-XLIV-4-W1-2020-159-2020.

[10] G. Zhou, S. Yuan, and S. Luo, "Mesh simplification algorithm based on the quadratic error metric and triangle collapse," *IEEE Access*, vol. 8, no. January 2020, pp. 196341–196350, 2020, doi: 10.1109/ACCESS.2020.3034075.

[11] S. Ramanathan, A. A. Kassim, and T. S. Tan, "Impact of vertex clustering on registration-based 3D dynamic mesh coding," *Image Vis. Comput.*, vol. 26, no. 7, pp. 1012–1026, 2008, doi: 10.1016/j.imavis.2007.11.005.

[12] H. Bai, T. Shen, L. Huo, X. Wang, and X. Liu, "Improved Edge Folding Algorithm for 3D Building Models Taking into Account the Visual Features," *Buildings*, vol. 13, no. 11, 2023, doi: 10.3390/buildings13112739.

[13] R. Riwinoto and A. Nugroho, "3D Optimization Analysis in Augmented Reality Application," 2024, doi: 10.4108/eai.7-11-2023.2342933.

[14] M. Ali, Z. Hussain, and M. S. Yang, "Hausdorff Distance and Similarity Measures for Single-Valued Neutrosophic Sets with Application in Multi-Criteria Decision Making," *Electron.*, vol. 12, no. 1, 2023, doi: 10.3390/electronics12010201.

[15] J. C. Xia, J. El-Sana, and A. Varshney, "AdAptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models," *IEEE Trans. Vis. Comput. Graph.*, vol. 3, no. 2, pp. 171–183, 1997, doi: 10.1109/2945.597799.

[16] P. Pellizzoni and G. Savio, "Mesh simplification by curvature-enhanced quadratic error metrics," *J. Comput. Sci.*, vol. 16, no. 8, pp. 1195–1202, 2020, doi: 10.3844/JCSSP.2020.1195.1202.

[17] B. Rodriguez-Garcia, I. Miguel-Alonso, H. Guillen-Sanz, and A. Bustillo, "LoDCalculator: A level of detail classification software for 3D models in the Blender environment," *SoftwareX*, vol. 30, no. December 2024, p. 102107, 2025, doi: 10.1016/j.softx.2025.102107.

[18] A. M. Boutsi, C. Ioannidis, and S. Verykokou, "Multi-Resolution 3D Rendering for High-Performance Web AR," *Sensors*, vol. 23, no. 15, 2023, doi: 10.3390/s23156885.