Automatic Level of Detail for Improved Rendering by CPU Usage

Rattanapol Joubjang* and Thanawin Rakthanmanon

Department of Computer Engineering, Kasetsart University, Bangkok, Thailand
*Corresponding author, E-mail: rattanapol.j@ku.th

Abstract

This research focuses on advancing gaming experiences by enhancing the management of the Level of Detail (LOD) for consistent Frames per Second (FPS). Rooted in the evolution of 3D computer graphics, the study introduces spatial hashing, integrated CPU/GPU data analysis, and adaptive LOD updates as a unified approach. The primary challenge involves dynamically managing LOD in game environments to balance visual fidelity and optimize performance. Traditional methodologies often fall short in dynamic scenes or diverse hardware environments. This study aims to address these challenges by seamlessly integrating spatial hashing, CPU/GPU data analysis, and adaptive LOD updates to redefine industry standards for LOD management and provide players with an immersive experience coupled with optimal performance. Furthermore, the study quantifies the impact of dynamic LOD management on gaming performance metrics, evaluates the effectiveness of adaptive LOD updates in real-time gaming environments, assesses the efficiency of spatial hashing in optimizing LOD selection processes, and determines the feasibility of integrating CPU/GPU data analysis for informed LOD decisions.

Keywords*: *Adaptive LOD, Rendering improvement, Model management, Data analysis, Spatial hashing

1. Introduction

In the ever-evolving landscape of gaming, advancements have been remarkable, spanning graphics, game engines, and software capabilities. Modern games demand robust hardware, with CPUs handling tasks like physical simulation, player input, and NPC interactions, while GPUs are tasked with rendering graphics based on CPU instructions. Central to this immersive experience is the concept of Level of Detail (LOD) management (Clark, 1976). LOD dynamically adjusts object detail based on viewer proximity, optimizing resource allocation. It renders distant objects with lower detail and reserves higher detail for those that are nearby. Balancing visual fidelity with performance remains a challenge. Traditional methods struggle in dynamic or densely populated settings.

In response to these challenges, this research explores an innovative approach to LOD management. This approach integrates spatial hashing, CPU/GPU data analysis, and adaptive LOD updates to set a new standard in LOD selection methodologies. Spatial hashing offers a solution, organizing the game world into a grid of cells and associating objects with cells based on spatial coordinates. This enables efficient queries for nearby objects, forming a robust foundation for LOD decisions. Integrating CPU and GPU data analysis has become increasingly vital. The CPU evaluates object significance, while the GPU assesses rendering workload. Combining these analyses yields more informed LOD decisions, considering both object importance and rendering efficiency.

Furthermore, the research introduces the concept of adaptive LOD updates. This dynamic system continuously monitors the game world for changes and adjusts LOD levels in real-time. This ensures that the

LOD selection remains optimized as the player navigates or objects move, thus preserving both performance and visual quality.

2. Objectives

1) Quantify the impact of dynamic LOD management on gaming performance metrics
2) Evaluate the effectiveness of adaptive LOD updates in real-time gaming environments
3) Assess the efficiency of spatial hashing in optimizing LOD selection processes
4) Determine the feasibility of integrating CPU/GPU data analysis for informed LOD decisions

3. Materials and Methods

Automatic LOD Improve Rendering by CPU usage is about how to improve the performance of any game that uses LOD. Automatic LOD is a method to calculate CPU usage and GPU usage and compare them with the performance of the game. By collecting CPU usage, fps, number of triangles, number of vertices, terrain data, textures, and computing to select the level of detail object, it is possible to reveal which one should be selected for the best output in the game and give players the best experience.

Gather datasets then create training and testing data. Prediction of Data transfer aims to manage LOD systems from the testing data. The model and game settings are calibrated for the user's computer.

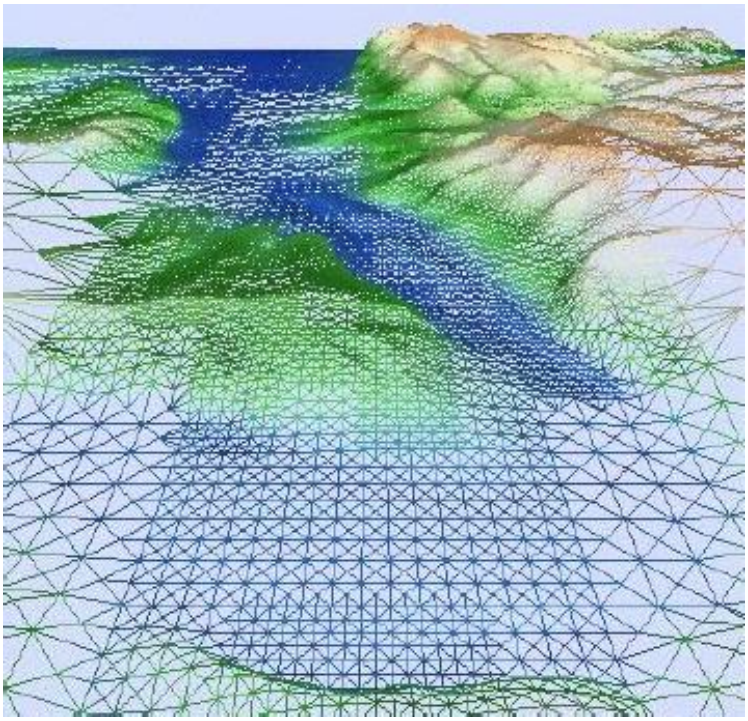

Figure 1: Terrain from the Ringlike Level of Detail in Real-Time Terrain Rendering (Wei, 2009). This shows how LOD adapts to terrain by reducing the mesh of the area that is far away in level of detail.

3.1 Generate Terrain

Start with an idea of how to create a predictor for Automatic LOD. It needs to gather data on the terrain. The data needed for this research include CPU usage, fps, GPU usage, and mesh (triangles amount)

by using the noise generation method to create the depth of the terrain, size of terrain, and quality in the use of texture (Amaris et al., 2016).

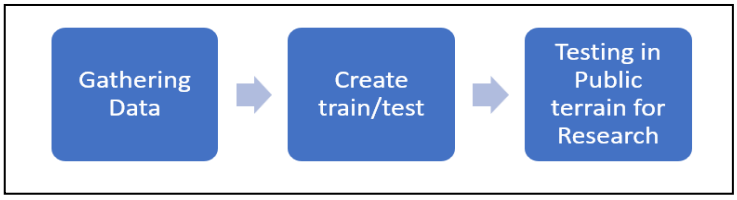

Figure 2: Research process

After creating the model by generating random terrain and gathering data, the model to create the predictor will be Multiple Linear Regression. This model must be used because of the data from the higher mesh; more fps will decrease so that it looks like Linear Regression, which will work out from this solution (Arafa et al., 2019). The regression equation that shows a linear relationship between a single dependent variable and one or more independent variables such as CPU usage, fps, and GPU usage is shown below:

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + ... + b_{k-1}x_{k-1} + b_kx_k$$

In this equation, ŷ is the prediction of the LOD, which is the level that should be selected. The value of the independent variable k is denoted as $x_1, x_2, x3,..., x_k$. Finally, we will get the model for prediction to check the solution. In the next process, we evaluate with the real one, such as the Unity Benchmark terrain test scene.

Unity Engine is used to track CPU usage and GPU usage while rendering data from benchmark datasets. After receiving the result, it needs to be calculated using linear regression and then compared with fps. The dataset will be models, CPU usage, GPU usage, LOD, and prediction from user behavior (Xiang-bin et al., 2008).

3.2 Object Distance

In computer graphics and interactive environments, accurately determining the distance between objects and the viewer is crucial, forming the basis for processes like Level of Detail (LOD) selection. This involves rendering objects with varying detail levels based on their proximity to the viewer. The theory of object distance relies on specialized algorithms, with the Euclidean distance formula being fundamental. This formula calculates the straight-line distance between two points in 3D space and is particularly vital for determining distances between the viewer and objects in a scene.

The change in frequency between objects and density that show on the viewport must be determined. This is done by storing every object in the scene using the feature of an LOD by calculating between the set of layers for each level. Starting from finding the distance between a set of layers for each level, if $a = (a_1, a_2, a_3)$ and $b = (b_1, b_2, b_3)$ are points on a sphere of radius r>0 centered at the origin of Euclidean 3-space, the distance from a to b along the surface of the sphere is:

$$d(a,b) = r\left(\frac{a \cdot b}{r^2}\right) = \left(\frac{a_1b_1 + a_2b_2 + a_3b_3}{r^2}\right)$$

To grasp this concept, contemplate the plane formed by points a and b, along with the origin. If θ is the angle between the vectors a and b, then $a \cdot b = r^2 cos\theta$, and the short arc joining a and b has length $r\theta$. By finding distance, it is now possible to sort by layer in level.

In computer systems with low floating-point precision, the spherical law of the cosine formula may have a large rounding error if the distance is small (If two points are 1 km apart on the earth's surface, the cosine of the center angle is approximately 0.99999999). For modern 64-bit floating-point numbers, the spherical law of the cosine formula is used. There is no serious rounding error for distances greater than a few meters on the surface.

3.3 Spatial Hashing for Object Organization

Spatial hashing is a pivotal technique in computer graphics and game development for efficiently organizing objects within a 2D or 3D space. It groups nearby objects, notably reducing computational complexity for tasks like collision detection, visibility culling, and Level of Detail (LOD) selection. The process begins with initializing a hash grid and then dividing the game world into cells associated with unique hash keys. Cell size can be adjusted for varying levels of detail, balancing granularity with memory and computation requirements. As objects move, they are mapped to respective spatial hash cells based on their positions, allowing for targeted computations and optimizations.

This technique optimizes object queries by restricting evaluations to objects in the same spatial hash cell, notably reducing the number of objects considered. In the context of automatic LOD selection, spatial hashing aids in CPU data analysis, allowing focus on relevant objects within the same cell. This prevents unnecessary computations for distant or irrelevant objects. Overall, spatial hashing is an efficient method for organizing and managing objects in a game world, which is crucial for operations requiring spatial querying and optimization.

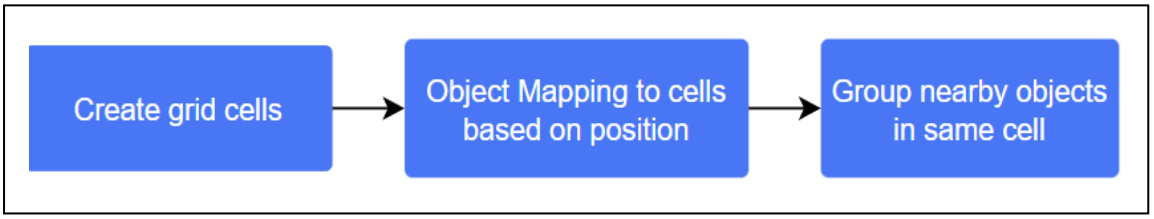

Figure 3: Object organization process

3.4 CPU and GPU Data Analysis for LOD Selection

CPU and GPU data analysis are integral to the automatic LOD selection process. This entails evaluating metrics to determine the appropriate LOD levels for objects. Firstly, metrics like geometric complexity and occlusion queries are computed in the CPU analysis phase for each object. This assesses their significance, guiding LOD-level decisions. Spatial hashing streamlines this process by allowing the CPU to focus on nearby objects, avoiding unnecessary computations for distant ones.

Subsequently, metrics like frame time and triangle count are assessed in the GPU analysis phase (Zhao & Ma, 2009) for rendering complexity. Spatial hashing aids by grouping objects with similar characteristics, optimizing data access. The integration of CPU and GPU data, facilitated by spatial hashing, ensures a comprehensive LOD selection process. Overall, these analyses, bolstered by spatial hashing, strike a balance between performance and visual quality.

3.5 Adaptive LOD update

Adaptive LOD updates are pivotal in the automatic LOD selection approach, dynamically adjusting LOD levels for objects as the game world evolves. This process continuously monitors the scene, identifying changes that may affect LOD requirements (Agus Eko Minarno, 2020). Through spatial hashing, it selectively updates affected areas, avoiding needless computations for unchanged parts of the scene, ensuring an efficient LOD adjustment mechanism.

The adaptive LOD update reevaluates object significance based on CPU metrics, prioritizing higher LOD levels for more significant or newly visible objects. Concurrently, GPU metrics determine rendering complexity, guiding effective LOD-level assignments for optimal GPU resource management.

The integration of updated CPU and GPU data through spatial hashing guarantees informed LOD selections. This dynamic adjustment process optimizes performance while preserving visual quality (Wei, 2009). In conclusion, the interplay of spatial hashing, CPU/GPU data analysis, and adaptive LOD updates achieves efficient automatic LOD selection, culminating in an enhanced gaming experience with stable FPS and visually compelling graphics.

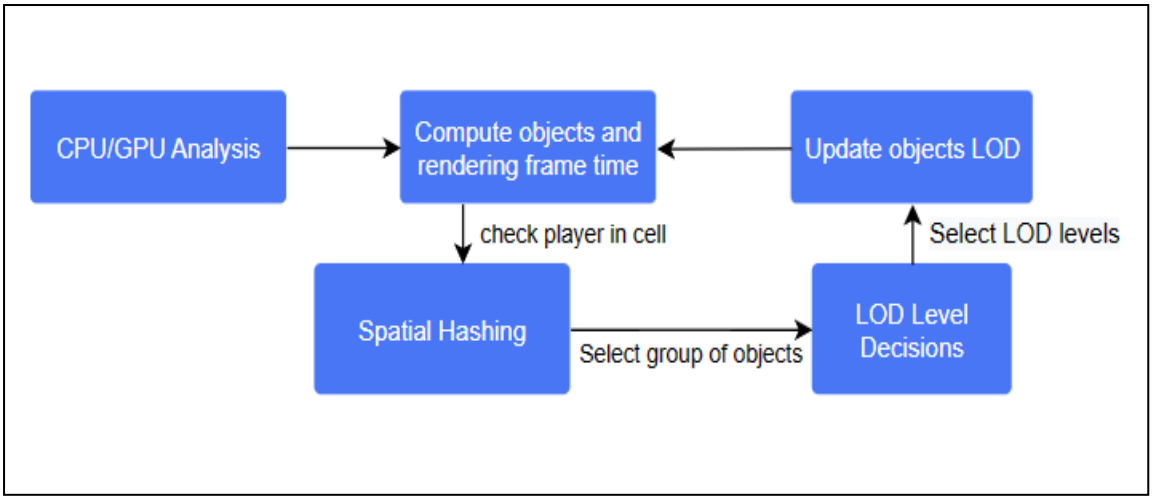

Figure 4: CPU/GPU Data Analysis for LOD Selection and updated LOD levels for objects processing

4. Results and Discussion

4.1 Results

Adaptive LOD extensive experiments were conducted using a variety of game scenes and scenarios to evaluate the effectiveness of the proposed approach. The experimental results demonstrated significant improvements in performance and visual quality compared to traditional manual LOD assignment and distance-based approaches. The adaptive LOD update process ensures a consistent and enjoyable gaming experience, even as the game world evolves.

Performance metrics included frame rate stability, rendering efficiency, and memory usage. The proposed approach consistently maintained a stable frame rate across diverse scenarios, minimizing frame rate drops during object transitions. Rendering efficiency was notably improved due to the integrated CPU and GPU data analysis, which optimized LOD levels based on object relevance and rendering complexity. Moreover, the approach demonstrated efficient memory usage by dynamically adjusting LOD levels and rendering only necessary object details.

Table 1: Test results for the 2 areas comparing two different areas before and after reducing the mesh of the objects

	Original	Improved LOD

	Area 1	Area 2	Area **1**	Area 2
CPU usage (millisecond)	15.1	10.8	14.2	10.3
Triangles (Million)	1.9	1.2	1.6	0.9
FPS	66.1	121	90	146

This method is easy to use and has high performance. Because you can get the results by using your environment hardware and adapting to this research, the benchmark that will be used for this research is the Unity Performance Benchmark, which is achieved by calculating from the camera that captures the object on the screen.

To establish the effectiveness of our approach, we compared it with traditional manual LOD assignment and distance-based methods. In scenarios with rapidly changing camera angles and object distances, the proposed approach outperformed traditional methods by preventing LOD popping and ensuring smooth transitions between levels of detail.

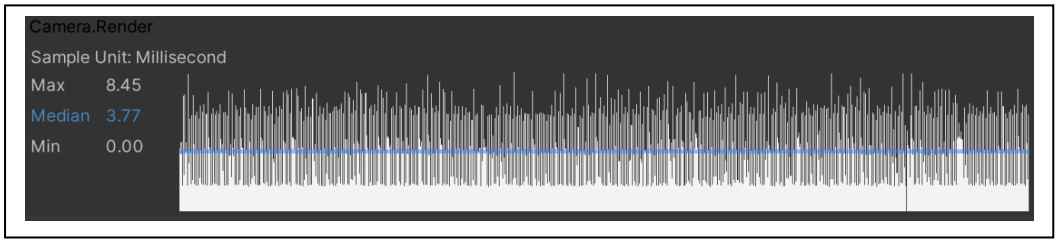

Figure 5: Results of Camera Rendering Performance; the result of the object in the scene is rendered in the camera.

After getting the object, it will check the number of vertices and triangles in objects with this data. It can check if the fps is in the maximum performance. If not, the method compares with CPU usage and then controls the LOD of the object that is rendered on the screen.

Our approach consistently delivered enhanced visual quality across different scenarios. Objects retained the appropriate level of detail regardless of camera distance, resulting in a more visually appealing and immersive gameplay experience. Textures, shadows, and finer object details were better preserved due to the adaptive LOD update mechanism facilitated by spatial hashing. The experimental results not only validated the efficacy of our approach but also highlighted its versatility. The proposed methodology proved its effectiveness across varying game scenes, displaying its adaptability to various levels of complexity and environments.

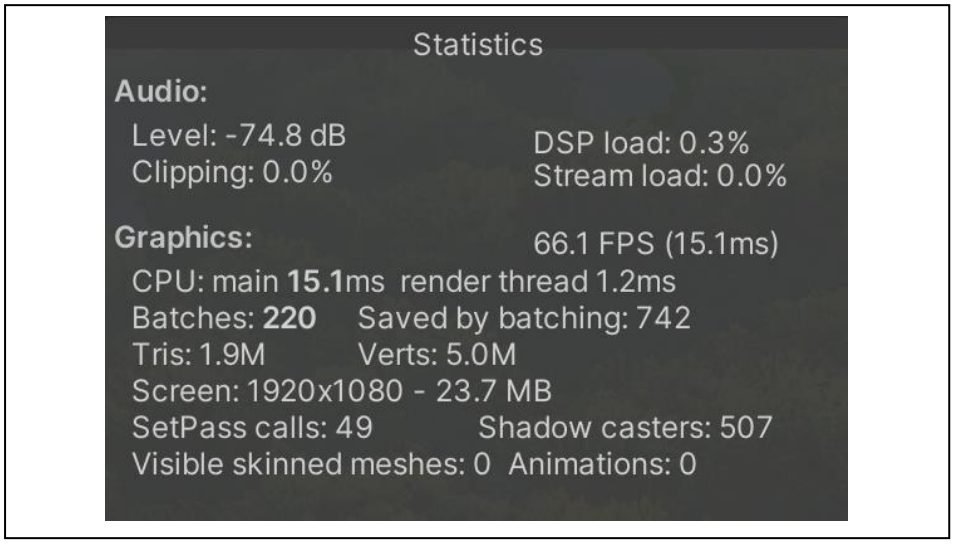

Figure 6: Results of Benchmark Test in Unity's play mode

When in Unity's play mode, this research method collects data from statistics, as in Figure 6, and then summarizes it with the users' settings, which is the fps target. As seen with the high Tris (triangles) amount, and with the current CPU usage drain FPS, the result will need to improve more than that. This result is from the Unity Example scene using the terrain with many trees. It does not have any characters, NPC, or heavy particles. Thus, this result is not satisfactory, and it is necessary to set up more suitable fps for such a small scene. It should run smoother and with more FPS.

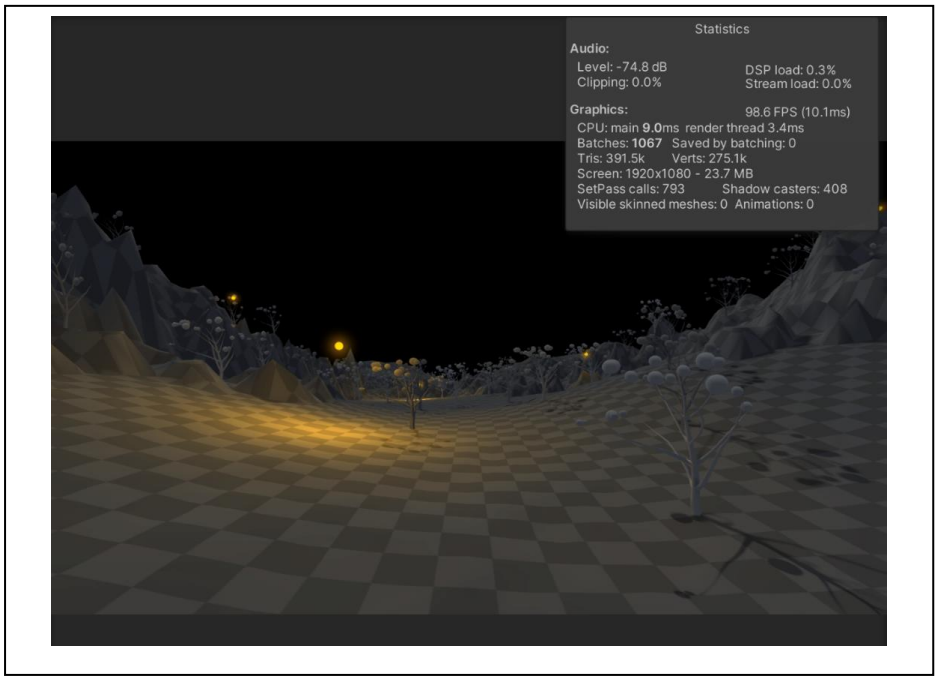

Figure 7: Results of Benchmark Test in Unity's play mode after calculating and setting up from the high-quality terrain with many objects; it reduces the mesh to improve FPS and rendering performance

Despite these achievements, potential limitations include the impact of object distribution and grid size on spatial hashing efficiency, requiring fine-tuning for optimal performance in different game scenarios. Overall, the proposed approach achieves an optimal balance between performance and visual quality in gaming applications with its efficient spatial hashing, seamless CPU/GPU data analysis integration, and responsive adaptive LOD updates. This research lays a solid foundation for the future of LOD selection, promising advancements in the gaming industry's standards for visual quality and performance.

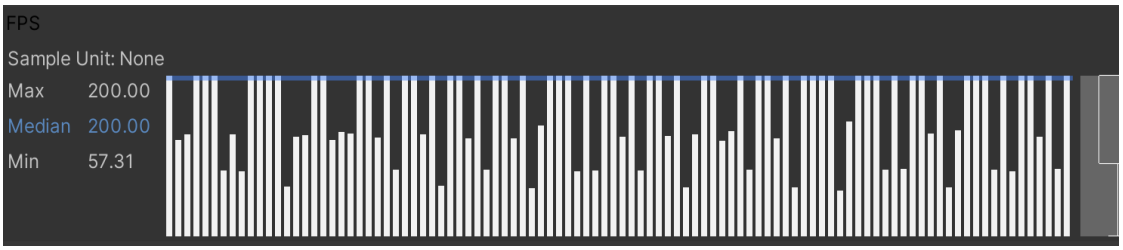

Figure 8: Test results for FPS when in Unity's play mode; the results need to be pushed on a threshold so the output will not show with noises

4.2 Discussion

The discussion and future work section delves into the implications of the research findings and proposes potential areas for further exploration and enhancement. The research successfully achieved its objectives by advancing gaming experiences through the integration of spatial hashing, CPU/GPU data analysis, and adaptive LOD updates, resulting in enhanced visual fidelity and optimal performance. However, a limitation lies in the requirement of varied hardware configurations, suggesting a need for standardization to facilitate broader implementation.

While the experimental results demonstrated the efficacy of the spatial hashing technique in optimizing object organization and LOD selection, fine-tuning the spatial hashing parameters remains an avenue for future exploration. The optimal cell size within the spatial hash grid could be determined through experimentation to strike the right balance between granularity and computational efficiency. This could lead to even more precise object grouping, potentially enhancing LOD transitions and overall scene optimization.

An intriguing avenue for future research involves the development of adaptive spatial hashing techniques. These techniques could dynamically adjust the grid size based on factors such as the density of objects, camera movement, and rendering workload. This adaptability could significantly improve performance in highly dynamic and crowded game environments, where the distribution of objects is not uniform. Implementing such adaptive spatial hashing would require sophisticated algorithms to dynamically optimize the grid structure in real time.

While the proposed approach integrates CPU and GPU data analysis for LOD selection, future work could explore hybrid strategies that combine various LOD selection methodologies. For instance, incorporating AI-driven techniques to predict player behavior and object importance could complement the CPU and GPU-based analysis. This could result in more accurate LOD decisions, particularly in scenarios where CPU and GPU metrics might not fully capture player preferences or specific gameplay dynamics.

5. Conclusion

The integrated approach of spatial hashing, CPU data analysis, and adaptive LOD updates marks a significant leap forward in LOD selection. By dynamically managing object distances and optimizing LOD

levels, the methodology achieves an optimal balance between visual fidelity and frame rate stability. Rigorous validation across diverse scenarios underscores the accuracy and effectiveness of the approach.

Additionally, the implementation of dynamic LOD management demonstrates a pronounced enhancement in gaming performance metrics, affirming its pivotal role in contemporary gaming environments. Furthermore, the seamless integration of CPU/GPU data analysis empowers informed LOD decisions, amplifying the efficiency of adaptive LOD updates. Overall, this comprehensive strategy not only elevates gaming experiences but also sets a promising precedent for future advancements in LOD optimization.

6. Acknowledgements

The authors would like to express their sincere gratitude to the computer engineering faculty, including Asst. Prof. Jittats Fakcharoenphol and Assoc. Prof. Punpiti Piamsa-Nga, for their invaluable support and guidance throughout this research. Their knowledge and advice were crucial in reaching this milestone. Lastly, the authors deeply appreciate the unwavering support of their parents and friends throughout this research journey.

7. References

Agus Eko Minarno, A. R. A., Wahyu Andhyka Kusuma, Wildan Suharso, Hardianto Wibowo. (2020). *Optimizing Game Performance with Dynamic Level of Detail Mesh Terrain Based on CPU Usage* 2020 3rd International Conference on Intelligent Autonomous Systems (ICoIAS), Singapore, Singapore. https://ieeexplore.ieee.org/document/9081835/

Amarís, M., de Camargo, R. Y., Dyab, M., Goldman, A., & Trystram, D. (2016, October). A comparison of GPU execution time prediction using machine learning and analytical modeling. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)* (pp. 326-333). IEEE. https://doi.org/10.1109/NCA.2016.7778637

Arafa, Y., Badawy, A.-H. A., Chennupati, G., Santhi, N., & Eidenbenz, S. (2019). PPT-GPU: Scalable GPU Performance Modeling. *IEEE Computer Architecture Letters*, *18*(1), 55-58. https://doi.org/10.1109/lca.2019.2904497

Clark, J. H. (1976). Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, *19*(10), 547–554. https://doi.org/10.1145/360349.360354

Wei, Z. (2009, July). Ringlike level of detail in real-time terrain rendering. In *2009 International Conference on Environmental Science and Information Application Technology* (Vol. 3, pp. 716-719). IEEE. https://doi.org/10.1109/ESIAT.2009.490

Xiang-bin, S., Fang, L., Ling, D., Siqin, B., & Xianmin, C. (2008, August). An algorithm of the level of detail terrain. In *2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing* (pp. 912-916). IEEE. https://doi.org/10.1109/SNPD.2008.34

Zhao, Y., & Ma, Y. (2009, April). A modified LOD terrain model based on quadtree algorithm. In *2009 International Joint Conference on Computational Sciences and Optimization* (Vol. 2, pp. 259-263). IEEE. https://doi.org/10.1109/CSO.2009.195