3D Terrain Rendering using WebGL

1st Shubhendra Gautam
IIITA
Prayagraj, India
IIT2021142@iiita.ac.in

2nd Saikat Sadhukhan
IIITA
Prayagraj, India
IIT2021261@iiita.ac.in

3 3d Vishal kumar
IIITA
Prayagraj, India
IIT2021196@iiita.ac.in

4th Chandan Kumar
IIITA
Prayagraj, India
IIT2021209@iiita.ac.in

5th Avaneesh Rav Vikram
IIITA
Prayagraj, India
IIT2021211@iiita.ac.in

***Abstract*—**
This report presents a study on advancing terrain rendering technology to achieve consistent performance and visual quality across different devices. Through a comprehensive literature review, challenges and knowledge gaps in terrain rendering were identified, motivating the development of novel optimization techniques and cross-platform solutions. The project aims to enhance rendering performance on both laptop and phone-like devices while integrating real-time data for improved environmental simulation and virtual exploration. Results demonstrate significant improvements in rendering quality and cross-platform compatibility, underscoring the project's achievements and contributions to the field.

I. Introduction

Terrain rendering technology plays a crucial role in various applications, including gaming, simulation, and geographical analysis. However, achieving consistent rendering performance across different devices poses significant challenges due to hardware constraints and platform variations. This report addresses these challenges by proposing novel optimization techniques and cross-platform solutions to enhance terrain rendering performance.

II. Literature Review

In the domain of real-time terrain rendering techniques essential for applications like Virtual Reality, games, and geological visualization, extensive research has focused on achieving interactive rendering rates. Techniques are broadly classified into two categories: polygon-based and voxel-based models.

Polygon-Based Rendering: Utilizes polygonal meshes and hierarchical structures like BSP trees or Quadtree to optimize rendering performance and manage Level of Detail (LOD). Shader-based rendering techniques enable dynamic tessellation and displacement mapping for realistic terrain representation. Voxel-Based Rendering: Operates on volumetric representations, extracting polygonal meshes from voxel grids using algorithms like Marching Cubes. Offers advantages in memory efficiency and adaptability to dynamic terrain modifications.

Tile-Based Terrain Division: The approach partitions the terrain into tiles, facilitating efficient management of rendering complexity and resource utilization on mobile devices. By dividing the terrain into manageable chunks, the technique enables dynamic adjustment of level of detail (LOD) to maintain optimal performance while navigating across the landscape. Dynamic Level of Detail (LOD) Computation: Each tile undergoes adaptive LOD computation, ensuring that the terrain is rendered with appropriate detail based on factors such as proximity to the viewer and available computational resources. This dynamic LOD adjustment minimizes rendering overhead and enhances speed by allocating computational resources efficiently. Adaptive Triangle Strip Rendering: To optimize CPU usage, the method employs adaptive triangle strip rendering for each tile, reducing redundant processing and improving rendering efficiency. By generating triangle strips adaptively, the technique minimizes computational overhead while maintaining visual fidelity, particularly on resource-constrained mobile devices. Crack Removal Technique: A key aspect of the proposed method is the inclusion of a technique for removing cracks on mesh boundaries, which can occur due to inconsistencies in LOD transitions. By addressing these rendering artifacts, the technique ensures seamless transitions between different resolution levels, enhancing the overall visual quality of the rendered terrain.

The introduction of this multi-resolution technique for terrain rendering on mobile devices represents a significant advancement in the field, offering a streamlined approach to handling rendering complexity while optimizing performance. By leveraging adaptive LOD computation, triangle strip rendering, and crack removal techniques, the method demonstrates promise in enhancing the efficiency and visual quality of real-time terrain rendering on mobile platforms.

A. Knowledge Gap

Rendering Discrepancies Across Devices: Literature indicates a potential challenge in achieving consistent rendering performance across different devices phone-like devices. Less Performance for Mobile Platforms: We found a gap in knowledge regarding the performance techniques required to achieve high-quality rendering on mobile platforms with limited resources compared to desktop counterparts. User Experience Considerations: Understanding the impact of rendering performance on user experience, particularly in mobile contexts where factors like battery consumption and device heating may affect usability. Cross-platform Compatibility: Addressing challenges related to cross-platform compatibility and ensuring seamless rendering experiences across different operating systems and hardware configurations.

III. Proposed Methodology

The methodology for setting up and rendering 3D graphics begins with establishing the scene, camera, and renderer to create the foundational environment. This involves defining the spatial layout, camera viewpoint, and rendering engine parameters to ensure proper visualization. Subsequently, 3D models are loaded into the scene using GLTFLoader, facilitating the importation of diverse assets such as terrain, objects, and characters. Lighting setup is then implemented, employing directional lighting techniques to illuminate the scene realistically and enhance visual depth. Keyboard controls are integrated to enable user navigation and zooming functionalities, providing users with intuitive interaction options. Additionally, touch controls are incorporated to cater to mobile devices, allowing users to interact with the scene using gestures and touch-based inputs. Finally, the scene is animated using requestAnimationFrame to create dynamic visuals and enhance user engagement through fluid motion and transitions. This methodology ensures the creation of immersive and interactive 3D graphics experiences across various platforms and input devices.

A. Language and library used

For the implementation details section, we utilized WebGL technology as the core framework for rendering 3D terrain across various devices, particularly focusing on mobile platforms. To streamline the development process and enhance rendering capabilities, we incorporated Three.js, a widely-used JavaScript library built on top of WebGL.

WebGL Technology: Serving as the foundational technology for our project, WebGL provided a robust low-level API for rendering interactive 3D graphics within any compatible web browser. This technology offered direct access to the GPU, enabling efficient rendering of complex scenes and geometries.

Three.js Integration: We seamlessly integrated Three.js into our project to simplify intricate WebGL operations and abstract away the complexities associated with direct WebGL programming. Leveraging the comprehensive features and functionalities of Three.js, we could effortlessly create and manipulate 3D scenes, geometries, materials, and lights. This integration not only accelerated the development process but also facilitated the implementation of advanced rendering techniques, enhancing the visual quality and interactivity of our 3D terrain rendering application.

IV. Result and Analysis

Initial Focus on Laptop Rendering: Started by prioritizing rendering improvements on laptop devices, achieving high-quality graphics and visual appeal. Subsequent Shift to Phone-Like Devices: After establishing success on laptops, directed efforts towards improving rendering on phone-like devices, aligning with the primary task of rendering enhancement across different devices. Successful Achievement of Primary Task: Successfully enhanced rendering on phone-like devices, fulfilling the primary objective of improving rendering capabilities across various devices. Exceeded Previous Research Standards: Surpassed the rendering performance benchmarks established in prior research, indicating substantial progress and improvement in rendering capabilities.

V. conclusion

In summary, our AGA project has been a fruitful journey marked by growth and adaptation. We've successfully implemented rendering optimizations for mobile devices and incorporated valuable feedback from our professor. These enhancements underscore our commitment to continuous improvement and innovation. As we conclude this phase, we're eager to apply these learnings in future projects.

VI. References

[1] Hu, L., Yan, H., Li, L., Pan, Z., Liu, X., Zhang, Z. (2021). MHAT: An efficient model-heterogenous aggregation training scheme for federated learning. Information Sciences, 560, 493-503.

[2] Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H. R., Albarqouni, S., ... Cardoso, M. J. (2020). The future of digital health with federated learning. NPJ digital medicine, 3(1), 119.

[3] Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., Gao, Y. (2021). A survey on federated learning. Knowledge-Based Systems, 216, 106775.

[4] Mammen, P. M. (2021). Federated learning: Opportunities and challenges. arXiv preprint arXiv:2101.05428.

[5] Li, L., Fan, Y., Tse, M., Lin, K. Y. (2020). A review of applications in federated learning. Computers Industrial Engineering, 149, 106854.

[6] Gou, J., Yu, B., Maybank, S. J., Tao, D. (2021). Knowledge distillation: A survey. International Journal of Computer Vision, 129, 1789-1819.

[7] Yang, Q., Liu, Y., Chen, T., Tong, Y. (2019). Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2), 1-19.

[8] Lai, F., Dai, Y., Singapuram, S., Liu, J., Zhu, X., Madhyastha, H., Chowdhury, M. (2022, June). Fedscale: Benchmarking model and system performance of federated learning at scale. In International Conference on Machine Learning (pp. 11814-11827). PMLR.

[9] Qu, Z., Lin, K., Li, Z., Zhou, J. (2021, May). Federated learning's blessing: Fedavg has linear speedup. In ICLR

2021-Workshop on Distributed and Private Machine Learning (DPML).

[10] Zhong, Z., Zhou, Y., Wu, D., Chen, X., Chen, M., Li, C., Sheng, Q. Z. (2021, May). P-FedAvg: Parallelizing federated learning with theoretical guarantees. In IEEE INFOCOM 2021-IEEE Conference on Computer Communications (pp. 1-10). IEEE.

[11] Sun, T., Li, D., Wang, B. (2022). Decentralized federated averaging. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(4), 4289-4301.

[12] Lin, T., Kong, L., Stich, S. U., Jaggi, M. (2020). Ensemble distillation for robust model fusion in federated learning. Advances in Neural Information Processing Systems, 33, 2351-2363.