|  |
| --- |
| MTU Kerry |
| Capturing, Editing and Merging Gaussian Splats to Create Novel 3D Environments |
| Academic Year:2023/2024  Programme Title: Computing with Games Development – MT803  Module Title: Final Year Project  Module Code: PROJ 81003  Lecturer’s Name: Claire Horgan |

|  |
| --- |
| STUDENT Billy Elsbury T00224562  12-9-2024 |

# Abstract

Gaussian Splatting, a rendering technique originating in the 1990s, has recently seen significant advancements, making it a possible alternative to traditional 3D rendering methods. This paper explores the potential and limitations of Gaussian Splatting for creating novel environments in various applications, including game development. The exploration of techniques for editing and merging Splats to generate unique and visually impressive 3D scenes. This paper examines the process of editing and merging Gaussian Splats and compares it with other emerging methods, such as Neural Radiance Fields (NeRF) and traditional photogrammetry. This research demonstrates how Gaussian Splatting can contribute to creating highly detailed and dynamic virtual environments.

# Introduction

Computer graphics, photogrammetry, 3D rendering, and game development are the main research areas relevant to this research project. A focus on applying Gaussian Splatting, a point-cloud-based 3D rendering technique, to create novel and visually compelling 3D environments.

Techniques for editing and merging Gaussian Splats are investigated, exploring their potential to alter Gaussian Splats to create unique 3D scenes for various applications, such as game development. The research aims to assess its strengths, limitations, and potential contributions to 3D content creation.

Contents

[Abstract 1](#_Toc183992217)

[Introduction 1](#_Toc183992218)

[Chapter 1. Literature Review 3](#_Toc183992219)

[1.1 Introduction to Gaussian Splatting 3](#_Toc183992220)

[From Video to Gaussian Splat 4](#_Toc183992221)

[1.2 Gaussian Splatting in 3D Content Creation 5](#_Toc183992222)

[Applications in Game Development 5](#_Toc183992223)

[Photogrammetry 5](#_Toc183992224)

[Gaussian Splatting & Photogrammetry 6](#_Toc183992225)

[Emerging Techniques: NeRF and Beyond 6](#_Toc183992226)

[1.4: Future Directions and Potential Improvements 7](#_Toc183992227)

[Chapter 2. Methodology 8](#_Toc183992228)

[2.1 Research Question 8](#_Toc183992229)

[2.2 Data Capture & Gaussian Splat Creation 8](#_Toc183992230)

[2.3 Performance & File Size Optimisation 8](#_Toc183992231)

[Chapter 3. Design 8](#_Toc183992232)

[3.1 Gaussian Splats in Game Engines 8](#_Toc183992233)

[3.2 Collision Handling 9](#_Toc183992234)

[3.3 Plugin Enhancement & Code Exploration 9](#_Toc183992235)

[Chapter 4. Implementation 9](#_Toc183992236)

[3.1 Prototype Development 9](#_Toc183992237)

[Conclusion 10](#_Toc183992238)

[References 11](#_Toc183992239)

# Chapter 1. Literature Review

## 1.1 **Introduction to Gaussian Splatting**

The core idea of Gaussian Splatting goes back as far as described by Lee Westover in his 1991 dissertation “Splatting”. Westover derived the term Splatting "from a non-technical description of the feed-forward volume-rendering process" (Westover, 1991). Westover used the mental image of a snowball hitting a wall to inspire the name for the process, as it makes a ‘Splat’ sound and leaves “its contribution across the wall” (Westover, 1991). This metaphor highlights the essence of the Splatting process, where each volumetric data point (Splat) contributes to the final image by spreading its effect over the image plane and obscuring previous points for that view.

Westover suggested Splatting as a response to limitations in traditional triangle rendering, which often produced artefacts with limited interactive viewing. Westover found traditional rendering methods that “coerced the volumetric data into line and surface primitives” ​(Westover, 1991) could produce artefacts during data processing that could be mistaken for features in the data.

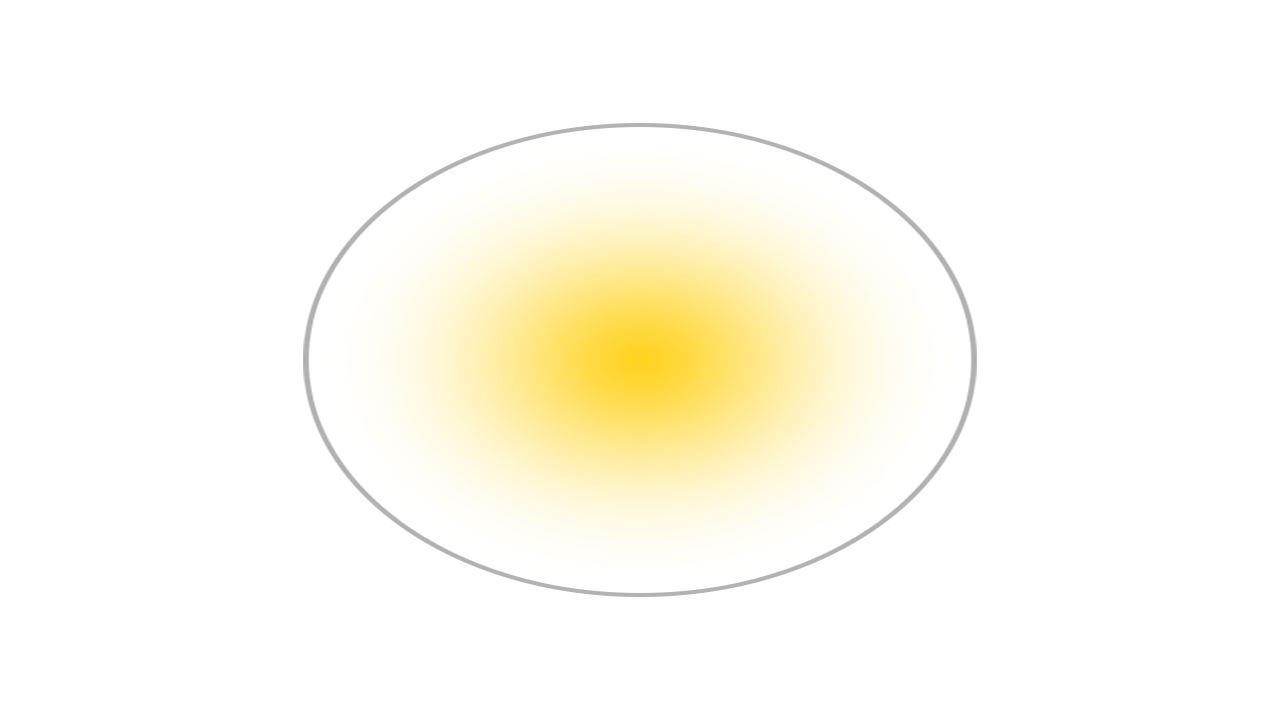
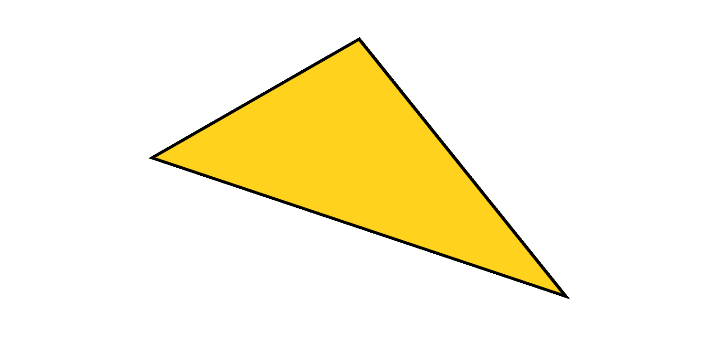


Figure 1 Rasterised Triangle & Gaussian representation (Ebert, 2023)

The core principles of Splatting revolve around its ability to map volume data onto an image plane without first transforming the data into geometric primitives. Gaussian functions are optimal for representing volume data due to their bell-curve-like (figure 2), which facilitates natural interpolation between data points.

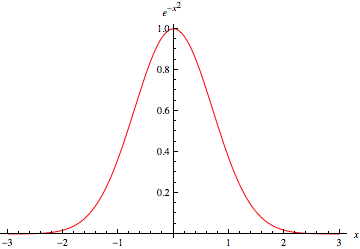


Figure 2 Gaussian Function (Weisstein, 2024)

One of the primary advantages of Gaussian Splatting is its computational efficiency. Using lightweight Gaussians Splats for scene representation enables efficient scene rendering, allowing real-time rendering of point cloud data with high visual quality (Kerbl et al., 2023). This novel approach to scene rendering allows for high visual fidelity while maintaining fast training times (Kerbl et al., 2023).

Despite the potential advantages of Gaussian-based rendering, Gaussian Splatting has limitations. The nature of using Splats for smooth blending of point cloud data can introduce artefacts during fast movement or view rotation and may struggle with representing high-frequency details (Radl et al., 2024). Additionally, while real-time rendering at high resolution is achievable, complex scenes may still face challenges maintaining speed and visual clarity at the highest quality levels.

### From Video to Gaussian Splat

The process of Gaussian Splatting in its current form primarily came from a 2023 paper, “3D Gaussian Splatting for Real-Time Radiance Field Rendering” (Kerbl et al., 2023). It was found that Neural Radiance Field (NeRF) rendering methods at the time were lacking in performance and produced noise in the final image. It was proposed that 3D Gaussians (Figure 2.0) be used to represent the point cloud data, retaining high visual fidelity while improving training times and performance with the aim of real-time rendering.

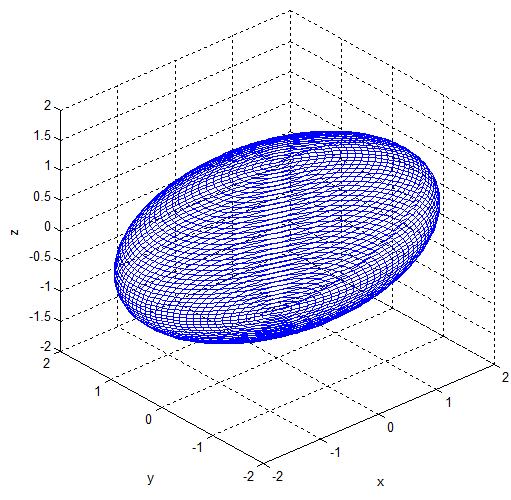


Figure 2.0: Visualisation of a 3D Gaussian model (Park et al., 2012).

The Gaussian Splatting process begins with the same inputs as other similar photogrammetry methods: the camera position between captured images is tracked with a technique such as “Structure-from-Motion” (SFM) (Snavely, Seitz and Szeliski, 2006). The sparse point cloud produced in the SFM process can be initialised with 3D Gaussians derived from the original captures. The properties of these 3D Gaussians can be optimised to fine-tune their position and opacity to best match the original capture; the visual fidelity is improved due to this sorting and blending optimisations. Additionally, anisotropic Splatting allows the Splats to represent the captured view from that angle best and improve accuracy when viewed from a novel perspective (Kerbl et al., 2023).

## **1.2 Gaussian Splatting in 3D Content Creation**

**Gaussian Splatting offers a promising alternative to traditional 3D rendering, with applications in game development, virtual environments, and interactive 3D content creation. As a volumetric rendering approach, Gaussian Splatting diverges from polygonal and mesh-based models, which dominate the field, offering unique benefits and constraints when used to generate complex 3D scenes.**

### **Applications in Game Development**

**Gaussian Splatting presents new possibilities for creating immersive and efficient environments in game development yet faces limitations in its early implementation stages. The core advantage lies in its ability to render point clouds as coherent scenes with minimal processing time. Gaussian Splatting could streamline rendering workflows for high-fidelity games by bypassing resource-intensive mesh generation steps. However, practical adoption is hampered by its current constraints: lack of dynamic lighting models, limitations in physics integration, limited tools for editing, and absence of Level of Detail (LOD) adaptations, which are critical for rendering performance in games.**

**The inability to re-light scenes dynamically comes from the Gaussian Splatting process using fixed, pre-computed lighting baked into the Splat colours; these colours can be adjusted to mimic lighting changes but do not accurately respond to lighting conditions, reducing utility in environments requiring dynamic lighting.**

### **Photogrammetry**

**Photogrammetry uses images to reconstruct detailed 3D models of real-world environments. The method of using photographs in measurement originated** as early as 1851 **from** Aimé Laussedat, **a French military engineer** (Albertz, 2007). Although Laussedat conducted experiments on hand-drawn images and later photographs for topographic mapping purposes, Laussedat called this method “Métrophotographie” (Albertz, 2007). Due to these initial experiments, Laussedat is often considered the “Father of Photogrammetry” (Albertz, 2007). There is, however, debate on how connected Laussedat’s experiments are to the development of photogrammetry as it is practised today (Polidori, 2020). The term photogrammetry was applied later by German architect Albrecht Meydenbauer in an 1867 paper entitled “Die Photogrammetrie” (Beelitz, 1867).

One of the first uses of photogrammetry in video game development is the 2014 game “The Vanishing of Ethan Carter” by an independent Polish studio (Statham, 2020). The technique was quickly adopted by larger studios (Statham, Jacob and Fridenfalk, 2020) as a complex but valuable method to create photoreal scenes for video games.

### **Gaussian Splatting & Photogrammetry**

**Gaussian Splatting shares similarities with photogrammetry in that both methods begin with camera-captured images and apply techniques like Structure-from-Motion (SfM)** (Snavely, Seitz and Szeliski, 2006) **to generate 3D data. Gaussian Splatting deviates in how it handles and renders this data. While traditional photogrammetry converts image data into meshes or voxel grids** (Snavely, Seitz and Szeliski, 2006)**, Gaussian Splatting employs low-weight Gaussian functions that allow smoother, more fluid transitions between data points** (Radl et al., 2024)**.**

**A notable difference lies in scalability and render speed: Gaussian Splatting’s lightweight nature allows faster rendering, making it better suited for real-time applications compared to photogrammetry’s polygon-heavy models, which can demand high processing power. This efficiency positions Gaussian Splatting as a potential solution for the increasing demand for video game quality and performance** (Atreya, 2022)**, as well as mobile and VR applications where performance constraints are a primary concern due to the limited hardware.**

### **Emerging Techniques: NeRF and Beyond**

**With advances in 3D content creation, methods like Neural Radiance Fields (NeRF)** (Mildenhall et al., 2020) **have gained traction for producing realistic and accurate 3D reconstructions. NeRF, which uses neural networks to predict the light intensity and colour across scenes, shares Gaussian Splatting’s goal of efficiently rendering point clouds but achieves this through deep learning. While NeRF can produce high-fidelity scenes, it requires significant computation and lengthy training times** (Mildenhall et al., 2020)**, which can limit its suitability for real-time applications.**

**Expand on NeRF a lot more what is it where/when did it come from etc**

**Gaussian Splatting offers a more immediate, lightweight solution with fewer computational demands than NeRF. However, Gaussian Splatting trades flexibility for fidelity due to the lack of meshed geometry. Both methods represent compelling alternatives to conventional 3D models, but each addresses distinct needs and constraints. Gaussian Splatting's real-time performance aligns well with applications that prioritise speed and fluidity over intricate detail, whereas NeRF may be preferable when flexible meshed models are required.**

## **1.3: Future Directions and Potential Improvements**

* **Research Gaps and Opportunities:** Identify areas where further research is needed to advance the application of Gaussian Splatting.
* **Potential Enhancements:** Suggest ways to improve Gaussian Splatting techniques, such as optimising performance or expanding its capabilities.
* **Integration with Other Technologies:** Explore the possibilities of combining Gaussian Splatting with other technologies, such as machine learning or artificial intelligence.

# Chapter 2. Methodology

## 2.1 Research Question

How can Gaussian Splats be edited and merged to create novel and visually impressive 3D environments for use in game development and beyond?

**Second semester, games-middleware assignment:** could a genetic algorithm be used with the NERF geometry of the scene to influence a set of colliders that would match the Gaussian Splat version of that scene? This would allow the visual fidelity of Gaussian Splats with the benefits of collisions for interactivity.

## 2.2 Data Capture & Gaussian Splat Creation

The process of capturing and creating Gaussian splats involves several key stages, beginning with data capture and proceeding through various tools and techniques to refine and finalise the Gaussian Splat creation.

Once video footage is captured, the footage can be processed using Luma AI (Lumalabs.ai, 2024) and Polycam (Poly.cam, 2024), both of which are web-based tools designed to convert 2D images or video into Gaussian Splats or NeRF representations. Luma AI and Polycam facilitate a streamlined, web-based, process for converting footage into Gaussian Splats.

For more customisation and refinement of the Gaussian Splats, Jawset Postshot (Jawset.com, 2024) can be used as a local, hardware-based solution. This tool allows for enhanced control over the creation of Gaussian Splats. Various settings can be specified before training, such as the number of images used, downsampling the resolution of the images, adjusting the number of features (3D points extracted), and determining the number of training steps. Postshot’s in-depth training settings improve the control over the quality of the outputted Gaussian Splat. Postshot allows for a more tailored result compared to the web-based automated solutions.

PICTURES

The original research paper on Gaussian Splatting (Pranckevičius, 2024) and its associated GitHub repository serves as a foundational resource for creating Gaussian Splats manually. The paper details the algorithms and best practices for creating Gaussian splats, ensuring that the generated splats meet the standards for the intended application. The repository provides sample code and implementation instructions that aid in the generation of high-quality splats.

This combination of high-quality video capture, automated processing through web tools, and manual refinement via hardware-based solutions ensures that the generated Gaussian splats are both high-fidelity and customisable, suitable for use in game development and further research.

## 2.3 Performance & File Size Optimisation

Gaussian Splats offer certain advantages in terms of visual fidelity when compared to traditional, high-resolution textured meshes used in video games. Due to their lightweight nature and efficient representation of 3D scenes, Gaussian Splats can achieve a high degree of visual realism with a lower computational load. However, to optimise their use within game engines, their file sizes should be reduced while maintaining an acceptable level of visual quality.

The file size of Gaussian Splats is primarily influenced by factors such as the number of splats, their resolution, and the quality settings used during the creation process. Larger numbers of high-quality Gaussian Splats result in increased file sizes, which can have a negative impact on performance and utility, particularly in real-time applications like video games. Reducing the resolution/number of Gaussian Splats or simplifying their representation can help mitigate these performance issues, but this comes at the cost of scene size and visual fidelity.

Optimising Gaussian Splats for use in game engines requires a careful balance between file size reduction and maintaining visual quality and scene complexity(the number of Gaussian Splats in a scene). Techniques such as downsampling, reducing the number of features (3D points) during the training process, or removing unnecessary Splats can be employed to achieve a better trade-off between performance and visual quality. Using a lower-quality version of a Gaussian Splat can result in faster rendering times and reduced memory usage, but may lose fine details such as subtle surface textures or small-scale features. Conversely, maintaining higher-quality splats ensures that fine details are preserved when editing, but this sacrifices performance, especially when large numbers of splats are used simultaneously.

Depending on the use case, such as blocking out a scene for later traditional asset creation, the relative loss in quality can be a valuable trade-off for the added performance and flexibility of smaller file-size Gaussian Splats.

Ultimately, the optimisation of Gaussian Splats involves balancing file size reduction with the quality required for their intended application. Regular testing and performance profiling are essential to ensure that optimisation strategies do not degrade the visual fidelity and utility of Gaussian Splat scenes while maximising computational efficiency.

## 3.1 Gaussian Splats in Game Engines

Process of importing the Gaussian Splats into Unity and Unreal Engine:

Unity Integration: Use of the UnityGaussianSplatting plugin, methods for manipulating and editing Splats. That plugin provides good tools for editing, moving splats, boxes and ovals to mass hide and remove splats, and adjusting transparency and splat size globally, as well as combining multiple splats into one which gives accurate occlusion of splats. Some editing can only be done when using the high-quality versions of splats which have reduced performance and larger file sizes, but a method used was to edit using the high quality then re-export and import using the lower quality import.

Unreal Engine Integration: Challenges and limitations encountered in Unreal Engine with fewer editing tools but leveraging relighting capabilities.

## 3.2 Collision Handling

Gaussian splats are not meshes and therefore do not have collisions. could a genetic algorithm be used with the NERF geometry of the scene to influence a set of colliders that would match the Gaussian Splat version of that scene? This would allow the visual fidelity of Gaussian Splats with the benefits of collisions for interactivity.

## 3.3 Plugin Enhancement & Code Exploration

Potential to modify or extend the [UnityGaussianSplatting](https://github.com/aras-p/UnityGaussianSplatting) plugin C# code in Unity.

Ability to edit individual Splat properties (colour, transparency, size).

# Chapter 3. Design

User Stories etc

# Chapter 4. Implementation

## 3.1 Prototype Development

The prototype development process for this project involved several iterative stages, starting with the capture and creation of Gaussian Splats, progressing through editing, merging, and refining these splats, and ultimately culminating in the construction of a demo prototype scene within Unity.

### Data Capture and Gaussian Splat Creation

Firstly, video footage of various locations was captured for the generation of Gaussian Splats. The footage was captured using a DJI Osmo Pocket 3 camera, high-resolution, stabilised footage of various objects was recorded during work placement at ACE. This equipment provided ideal filming conditions, including high resolution and a gimbal lock-on feature that allowed for smooth, stable orbits around the objects. The footage was then processed using Luma AI, a web-based tool for creating Gaussian Splats (Lumalabs.ai, 2024). Luma AI provided a convenient way to generate 3D point clouds that could be exported and used within game engines. Jawset Postshot, a local hardware-based solution, was utilised to produce more refined splats by leveraging its editable Gaussian splat-generating features, providing more customisable results. (Jawset.com, 2024).

### Importing and Editing in Unity

Once created, the Gaussian Splats were imported into Unity using the UnityGaussianSplatting plugin (Pranckevičius, 2024). This Unity plugin provided tools for importing Gaussian Splat ply files in different qualities which result in different file sizes. The plugin has tools for manipulating the imported Gaussian Splats, such as the ability to scale, move and delete individual or collections of Splats, as well as edit the size and transparency of all the Gaussian Splats in the scene. The plugin provides the utility to combine multiple splats into one, ensuring accurate occlusion of splats during rendering.

### Merging and Adjusting Splats

The process of merging Gaussian Splats involved strategically placing and aligning the various Gaussian Splats created from different sources, such as Gaussian Splats created from real video footage and Splats created with recordings from video game environments. This required a detailed iterative approach to ensure the splats blended cohesively within the scene. The adjustment process also involved cutting down many of the Splats from each capture, which helped to balance the visual integration of the Splats with each other. This iterative process was essential to refining the overall look and ensuring that the edited Gaussian Splats interacted in interesting ways in the Unity Demo Scene.

### Testing and Iteration

Throughout the prototype development process, regular testing was conducted to evaluate the performance and file sizes of the Gaussian Splats in the Unity scene. The primary metrics for testing included frame rate performance (FPS) and the Gaussian Splat PLY file size (Megabytes). Initial analysis showed that very large numbers of splats, primarily when used at high quality, led to noticeable performance drops. As a result, optimisations were made to reduce perfomance impact anf file size without significantly affecting the visual quality.

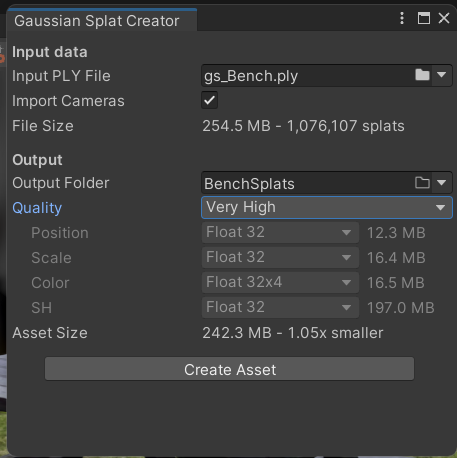


Figure 3: Very High Import Setting

A screenshot of a computer

Description automatically generated

Figure 4: Very Low Import Setting

A screenshot of a computer

Description automatically generated

Figure 5: Gaussian Splat Comparison

Figure 5 displays the same Gaussian Splat imported into Unity using a high-quality preset (Left) and a low-quality preset (Right) for comparison purposes. Visually the low-quality Gaussian Splat can be seen to be less accurate and lacks some finer details, especially on the surface of the table. However, the high quality Splat is also 17 times the file size at 242MB compared to the low quality at 14MB. The majority of this file size comes from the SH option, which stands for Spherical Harmonics. In the context of Gaussian Splatting, “the directional appearance component (color) of the radiance field is represented via spherical harmonics (SH)” (Kerbl et al., 2023).

With the high-quality import settings, each SH coefficient is stored as a highly precise 32-bit floating-point number, which retains accurate details in radiance and lighting. Accurate radiance representation helps define how the splat's colour and brightness vary based on the viewing angle and creates a more accurate Gaussian Splat.

With the low-quality import settings each SH coefficient is stored using a clustered quantisation approach, grouping similar SH coefficients into clusters. Using the cluster 4K option for SH import, the coefficients are heavily compressed, leading to a noticeable loss of detail in Splat accuracy. This loss of quality can be seen in the surface of the bench in Figure 5.

A screenshot of a computer

Description automatically generated

Figure 6: High Quality Import Perfomance Metrics

A screenshot of a computer

Description automatically generated

Figure 7: Low Quality Import Perfomance Metrics

The performance impact of importing Gaussian Splats with high and low-quality settings is illustrated in Figure 6 and Figure 7 by the difference in the number of frames per second (FPS) that are being rendered in the Unity game view. These tests were conducted on Unity 2022.3.47f1 using a system with an AMD RX 6950 XT GPU, a Ryzen 7 5600X CPU, and 32 GB of RAM. When using high-quality import settings for the same Gaussian Splat, the FPS averaged 280, compared to an average of 400 FPS for the low-quality import. Although there were no specific performance optimisations used for these tests beyond cropping and adjusting import quality, the performance metrics indicate the importance of considering the scene's performance when using Gaussian Splats. Careless importing and excessive use of multiple Gaussian Splats can lead to a significant decrease in performance.

The lower FPS associated with the high-quality import reflects the increased computational demand caused by the detailed radiance data stored in 32-bit floating-point SH coefficients. The precision of these coefficients allows for a more accurate representation of each individual Splat, contributing to the enhanced visual fidelity of the Gaussian Splat. However, this comes at the cost of higher file sizes and greater processing requirements.

In contrast, the low-quality import utilises clustered quantisation, which reduces the computational workload by compressing the SH coefficients. This simplification increases FPS but reduces some finer visual details, as seen in the surface textures and shading accuracy on the top of the bench in Figure 5 and can be seen as well in the outer grass in Figure 6 and Figure 7.

While the observed performance metrics are satisfactory on this hardware for scenes with a single Gaussian Splat, performance considerations will become increasingly critical as more Splats are imported, merged, and edited within the Unity scene. This consideration extends to lower-grade hardware performance; the combined file size and computational requirements of multiple high-quality Splats could lead to notable performance degradation. To maintain an FPS above 60, the optimal minimum for interactive real-time applications (Ionos.com, 2023), careful management of Splat quality, scene complexity, and optimisation techniques is necessary.

This analysis highlights the trade-off between visual fidelity and performance when using Gaussian Splats, reinforcing the importance of tailoring import settings to the application's requirements.

To mitigate performance issues, editing was done on Gaussian Splats using high-quality import settings then the Splats could be exported and re-imported using lower-quality settings. This workflow optimised performance and file sizes while allowing the use of all the editing tools without significantly compromising visual fidelity.

# Conclusion

# References

Albertz, J., 2007. A Look Back; 140 Years of Photogrammetry. *PHOTOGRAMMETRIC ENGINEERING*.

Atreya, S., 2022. *THE EVOLUTION OF VIDEO GAME GRAPHICS AND THEIR IMPACT ON THE INDUSTRY - ProQuest*. [online] Available at: <https://www.proquest.com/openview/ec5dd0d458105086072444bd3ba74239/1?pq-origsite=gscholar&cbl=2035897> [Accessed 2 December 2024].

Beelitz, C., 1867. Die Photogrammetrie. *Wochenblatt des Architektenvereins zu Berlin*, No. 49, p.1.

Ebert, D., 2023. *Introduction to 3D Gaussian Splatting*. [online] Available at: <https://huggingface.co/blog/gaussian-splatting> [Accessed 27 October 2024].

Ionos.com, 2023. *Frames per second (FPS) in TV, cinema, and gaming*. [online] IONOS Digital Guide. Available at: <https://www.ionos.com/digitalguide/server/know-how/fps/> [Accessed 8 December 2024].

Jawset.com, 2024. *Jawset Postshot*. [online] Available at: <https://www.jawset.com/> [Accessed 2 December 2024].

Kerbl, B., Kopanas, G., Leimkühler, T. and Drettakis, G., 2023. *3D Gaussian Splatting for Real-Time Radiance Field Rendering*. https://doi.org/10.48550/arXiv.2308.04079.

Lumalabs.ai, 2024. *Luma AI - Interactive Scenes*. [online] Luma AI - Interactive Scenes. Available at: <https://lumalabs.ai/interactive-scenes> [Accessed 2 December 2024].

Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R. and Ng, R., 2020. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. Available at: <http://arxiv.org/abs/2003.08934> [Accessed 27 October 2024].

Park, J.-H., Shin, Y.-D., Bae, J.-H. and Baeg, M.-H., 2012. Spatial Uncertainty Model for Visual Features Using a Kinect (TM) Sensor. *Sensors (Basel, Switzerland)*, 12, pp.8640–62. https://doi.org/10.3390/s120708640.

Polidori, L., 2020. ON LAUSSEDAT’S CONTRIBUTION TO THE EMERGENCE OF PHOTOGRAMMETRY. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B2-2020, pp.893–899. https://doi.org/10.5194/isprs-archives-XLIII-B2-2020-893-2020.

Poly.cam, 2024. *Polycam - LiDAR & 3D Scanner for iPhone & Android*. [online] Available at: <https://poly.cam/> [Accessed 2 December 2024].

Pranckevičius, A., 2024. *aras-p/UnityGaussianSplatting*. Available at: <https://github.com/aras-p/UnityGaussianSplatting> [Accessed 2 December 2024].

Radl, L., Steiner, M., Parger, M., Weinrauch, A., Kerbl, B. and Steinberger, M., 2024. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Trans. Graph.*, 43(4), p.64:1-64:17. https://doi.org/10.1145/3658187.

Snavely, N., Seitz, S.M. and Szeliski, R., 2006. Photo tourism: exploring photo collections in 3D. In: *ACM SIGGRAPH 2006 Papers*, SIGGRAPH ’06. [online] New York, NY, USA: Association for Computing Machinery. pp.835–846. https://doi.org/10.1145/1179352.1141964.

Statham, N., Jacob, J. and Fridenfalk, M., 2020. Photogrammetry for Game Environments 2014-2019: What Happened Since The Vanishing of Ethan Carter. In: *Proceedings of DiGRA 2020 Conference: Play Everywhere*. [online] Proceedings of DiGRA 2020 Conference: Play Everywhere. . Available at: <https://dl.digra.org/index.php/dl/article/view/1225> [Accessed 27 October 2024].

Statham, W., 2020. Use of Photogrammetry in Video Games: A Historical Overview. *Games and Culture*, 15(3), pp.289–307. https://doi.org/10.1177/1555412018786415.

Weisstein, E.W., 2024. *Gaussian Function*. [Text] Available at: <https://mathworld.wolfram.com/GaussianFunction.html> [Accessed 2 October 2024].

Westover, L.A., 1991. SPLATTING: A Parallel, Feed-Forward Volume Rendering Algorithm.