Hybrid Relightable 3D Gaussian Rendering

Design Document

Senior Design May 2025 Team 40
Clients: Jackson Vanderheyden & Brian Xicon
Advisor: Dr. Simanta Mitra
Team Members:
Luke Broglio,
Ethan Gasner,
Kyle Kohl,
Jackson Vanderheyden,
Brian Xicon,
sdmay25-40@iastate.edu
https://sdmay25-40.sd.ece.iastate.edu

Executive Summary

This project, the Hybrid Relightable 3D Gaussian Rendering, is designed to create high-quality 3D models using machine learning from videos of everyday objects. The user can then use it in 3D modeling spaces like games. We wanted our project to be used by anyone without specialized equipment and to be available to everyone to download our program. The goal was for the user just to take a video of an object with their phone and send it to their Unity project. After running our program, it should have a high-quality 3D Gaussian model of the object. Our program was to split our project into two parts: the machine learning part and the graphics path tracer part. The machine learning part was broken down into Structure-For-Motion and the Optimizer. The Structure-For-Motion code took the video of an object and transformed it into Gaussian points in the form of a ply file that was then passed into the Optimizer code. The Optimizer code took the cloud of Gaussian points and extracted data from the points using a PyTorch ML model. This data was then passed to the Unity path tracer code as a .ply file. With this information, the Unity side could create a 3D Gaussian object to raytrace. Our design meets the standard need for high-quality 3D Gaussian models. The requirements for anyone running our program would be having Unity installed and an NVIDIA graphics card to run the ML. With these two requirements, it doesn't reach everyone, but we believe anyone who would like to use our program would most likely meet the requirements already. The following steps of our projects would be to increase performance, get it to run faster and produce higher quality 3D models, and add more features to the scope.

Learning Summary

Development Standards & Practices Used

Standards:

- IEEE/ISO/IEC 12207-2017
- ISO/IEC 23053:2022
- ISO/IEC 23488:2022

Practices:

- Agile workflow
- Git
- Regular Sprint Reports

Summary of Requirements

- The system shall input a Structure from Motion (SfM) point cloud for the Gaussian model generator. Rationale: Gaussian optimizers initialize from and require a point.
- The system shall take a Gaussian model, which is composed of position, opacity, anisotropic covariance, and spherical harmonics, as input for the 3D Gaussian parser. Rationale: This format is required for Gaussian parameters.

- The system shall input a 3D Gaussian model for the Axis-Aligned Bounding Box (AABB) Bounding Volume Hierarchy (BVH) generator.
- The system shall take Unity mesh objects as input for the AABB BVH generator.
- The system shall take 3D Gaussian models with constructed AABBs, triangle meshes with constructed AABBs, and camera parameters as input for the hybrid Gaussian and triangle mesh ray tracer. Rationale: All data is required to render a scene accurately
- The system shall take the final rendered texture from the hybrid Gaussian triangle mesh ray tracer as input for the Unity Camera's rendered texture. Rationale: We write to a texture and blit it to the screen to override Unity's default camera render pipeline.
- When rendering the next frame, the system shall place all paths into the ended paths buffer. Rationale: primary path generation operates on all ended paths.
- When the primary ray generation compute shader is called, and the ended paths buffer contains paths, the system shall generate a new origin and ray direction to align the path with its associated pixel for each path in the ended paths buffer. Rationale: Initialize all paths to start at the camera's origin and direction.
- When the determined path intersection is computed shader is called, and the ended paths buffer contains paths. The system shall calculate ray intersections given the path direction. If the path hits a Gaussian model or a triangle mesh, add the path to the continuing paths buffer; otherwise, add the path to the ended paths buffer.
- When the lighting calculation computes the shader is called, and the continuing paths buffer contains paths, the system shall determine how much light the surface absorbs at the intersection point and calculate the reflection direction. If the path has reached its max bound limit, don't let it continue.
- Initial State: A video or series of images is used as input to the SfM point cloud generator.
- Intermediate State: SfM point cloud generator produces a sparse point cloud correlating to object color and geometry. This is used as input to the Gaussian optimizer.
- Final State: Gaussians are initialized at point positions with default normals; PyTorch Gaussian optimizer shall adjust the normals and other parameters to align with the lighting environment.
- The system should be available as a package for the Unity Game Engine. It should be able to be added to existing Unity projects and work with the existing tools provided by Unity.
- The system should be easy to use with the tools and systems provided by the Unity engine and integrate naturally with the rest of the system.
- The system should be intuitive and self-explanatory, so it's usable by someone without a technical or graphics programming background.

- The system should run in real-time. This means that the system should allow the user to change the scene's lighting or the location of the camera's angle, and the system should update the scene dynamically at 30 FPS.
- The system should produce high-quality renders that don't appear fake to the user, i.e., renders don't contain glaring visual artifacts like noise, shadow ripples, or erroneous sampling.
- The system should be available for download as a Unity package using Unity's package manager.

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents apply to your project.

- Com S 3360
- Com S 3090
- Com S 4370
- Com S 2280
- Com S 4740
- Com S 4170
- ENG 3140
- Com S 3270
- Math 2070

New Skills/Knowledge Acquired That Were Not Taught in Courses

- Compute Shaders
- Raytracing
- Python Scripting
- Unity Asset Store
- Gaussian Splatting
- Physically Based Rendering

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Problem Statement

Creating realistic 3D models of real-world objects using 3D modeling software presents significant challenges, including accurately replicating materials and topology. This process often requires tens to hundreds of hours of work from highly skilled graphic designers. Our system will enable users to generate realistic 3D Gaussian models directly from a video, seamlessly integrated into the Unity Game Engine.

The technique of generating images of a scene from any perspective using only a video or a series of photographs is known as novel view synthesis. Previously, this was done using Neural Radiance Fields (NeRFs). This method has costly training and rendering times, can produce subpar results, and cannot be combined with traditional polygon meshes.

Our solution is to create a novel view synthesis system using a new technique called 3D Gaussian Splatting (3DGS). This solution is significantly faster and creates higher-quality renders than NeRFs, but one lingering issue remains. Standard 3DGS models are not compatible with industry-standard polygon meshes.

To solve these issues, we built off the work of Relightable Gaussians and created a ray-traced rendering system that seamlessly integrates triangle and Gaussian models. This system will allow users to change the scene's lighting in real-time, take advantage of global illumination features such as reflections and shadows, and add traditional polygon meshes into the scene.

This system will be beneficial in various circumstances, including game development and 3D art, as part of sales to allow customers to see an accurate and high-quality depiction of a product, and as an improvement for current NeRFs or photogrammetry systems.

1.2 Intended Users

We believe three primary personas accurately represent the individuals who will use our product: “NeRF User,” “Non-Technical User,” and “3D Artist.”

The “NeRF User” persona actively performs novel view synthesis using existing technology such as NeRFs. This user group comprises people familiar with 3D rendering technology and novel view synthesis. “NeRF Users” seek better-performing view synthesis techniques to train and efficiently generate high-quality renders to improve their workflow. Our solution appeals to this group because of our system’s performance improvements over their current tools. It also offers new functionality, allowing them to do what they already do faster and with a higher-quality end product.

The second persona, the “Non-Technical User,” wants to display a scene or an object but cannot due to the technical knowledge barrier for current view synthesis models. For example, a “Non-Technical User” may be a realtor who wants to allow potential home buyers to view a property they have a video of as a 3D render, or they could be an owner of a furniture store wanting to visualize their products for their customers better. This user group must easily create and display an object or scene in three-dimensional space. Our product aims to be a user-friendly solution to novel view synthesis to ensure an enjoyable user experience for technical and non-technical users, making it incredibly appealing to “Non-Technical Users.”

The last user persona is the “3D Artist”. This person is the user who already has some knowledge of 3D rendering and is creating traditional triangle meshes using 3D modeling software such as Blender, Cinema4D, or Maya. While this user will already know about 3D rendering, they may not know of novel view synthesis. Currently, “3D artists” must recreate objects or scenes by hand, which is incredibly time-consuming. These users seek new technologies to improve their workflow that won’t inhibit current industry norms. To best accommodate this user group, we want our solution to be easily integrated with existing 3D rendering technologies and tools and usable by individuals without preexisting novel view synthesis knowledge. Our solution will directly appeal to this type of user by allowing them to efficiently create models and scenes from videos rather than making them by hand.

2 Requirements, Constraints, And Standards

2.1 Requirements & Constraints

2.1.1 Ubiquitous Functional Requirements

- The system shall input a Structure from Motion (SfM) point cloud for the Gaussian model optimizer. Rationale: Gaussian optimizers are initialized from and require a point cloud (constraint)
- The system shall take a Gaussian model, which is composed of position, opacity, anisotropic covariance, and spherical harmonics, as input for the 3D Gaussian parser. Rationale: These parameters are required to quantize Gaussians. (constraint)
- The system shall input a 3D Gaussian model for the Axis-Aligned Bounding Box (AABB) Bounding Volume Hierarchy (BVH) generator. (constraint)
- The system shall take Unity mesh objects as input for the AABB BVH generator. (constraint)
- The system shall take 3D Gaussian models with constructed AABBs, triangle meshes with constructed AABBs, and camera parameters as input for the hybrid Gaussian and triangle mesh ray tracer. Rationale: all data is required to accurately render a scene (constraint)
- The system shall take the final rendered texture from the hybrid Gaussian triangle mesh ray tracer as input for the Unity Camera’s rendered texture. Rationale: We write to a texture and blit it to the screen to override Unity’s default camera render pipeline. (constraint)

2.1.2 Event-Driven Functional Requirements

- When rendering the next frame, the system shall place all paths into the ended paths buffer. Rationale: primary path generation operates on all ended paths.
- When the primary ray generation compute shader is called, and the ended paths buffer contains paths, the system shall generate a new origin and ray direction to align the path with its associated pixel for each path in the ended paths buffer. Rationale: Initialize all paths to start at the camera’s origin and direction.

- When the determined path intersection is computed shader is called, and the ended paths buffer contains paths. The system shall calculate ray intersections given the path direction. If the path hits a Gaussian model or a triangle mesh, add the path to the continuing paths buffer; otherwise, add the path to the ended paths buffer.

- When the lighting calculation computes the shader is called, and the continuing paths buffer contains paths, the system shall determine how much light the surface absorbs at the intersection point and calculate the reflection direction. If the path has reached its max bound limit, don't let it continue.

2.1.3 State-Driven Functional Requirements

- Initial State: A video or series of images is used as input to the SfM point cloud generator.

- Intermediate State: SfM point cloud generator produces a sparse point cloud correlating to object color and geometry. This is used as input to the Gaussian optimizer.

- Final State: Gaussians are initialized at point positions with default normals; PyTorch Gaussian optimizer shall adjust the normals and other parameters to align with the lighting environment.

2.1.4 Look & Feel Nonfunctional Requirements

- The system should be available as a package for the Unity Game Engine. It should be able to be added to existing Unity projects and work with the existing tools provided by Unity. Rationale: Using the built-in asset store framework greatly improves usability. (constraint)

2.1.5 Usability Nonfunctional Requirements

- The system should be easy to use with the tools and systems provided by the Unity engine and integrate naturally with the rest of the system.

- The system should be intuitive and self-explanatory, so it's usable by someone without a technical or graphics programming background.

2.1.6 Performance Nonfunctional Requirements

- The system should run in real-time. This means that the system should allow the user to change the scene's lighting or the location of the camera's angle, and the system should update the scene dynamically at 30 FPS (constraint)

- The system should produce high-quality renders that don't appear fake to the user, i.e., renders don't contain glaring visual artifacts like noise, shadow ripples, or erroneous sampling.

2.1.7 Operational Nonfunctional Requirements

- The system should be available for download as a Unity package using Unity's package manager. Rationale: Using the built-in asset store framework greatly improves usability. (constraint)

2.2 Engineering Standards

Several key frameworks may apply when considering engineering standards for our project, including IEEE/ISO/IEC 12207-2017, ISO/IEC 23053:2022, and ISO/IEC 23488:2022. First, standard IEEE/ISO/IEC 12207-2017 outlines a comprehensive framework for software lifecycle processes, detailing best practices for software development, maintenance, and management. This standard ensures well-defined software processes, promoting consistency and quality throughout the development lifecycle. Following this standard will help our team ensure that our project constantly evolves and improves weekly.

We will integrate the IEEE/ISO/IEC 12207-2017 framework into our development process. This will include establishing precise requirements, design, implementation, testing, and maintenance throughout development. We will also conduct regular reviews and team updates via the agile development process. This will ensure that we adhere to the standard through all stages of development.

ISO/IEC 23053:2022 provides guidelines for assessing ML systems that use machine learning tactics, focusing on their reliability and performance. As ML plays a vital role in many projects, following this standard is crucial to ensure that the systems developed are effective and trustworthy. By implementing the principles in this standard, our team can better evaluate the robustness of our Gaussian point optimizer and our material prediction machine learning models; this will enhance user confidence in our final deliverable.

To comply with standard ISO/IEC 23053:2022, we plan to develop a comprehensive evaluation for our generated machine learning models, specifically the Gaussian point optimizer and the material prediction models. When creating our evaluation framework, we will use performance metrics to easily allow us to check the reliability and effectiveness of our models. We will perform these metrics throughout the model's development cycle.

Finally, ISO/IEC 23488:2022 focuses on representing objects and environments for image-based rendering in virtual, mixed, and augmented reality (VR/AR). This standard provides clear rules for capturing and showing the visual details of real and virtual objects, which helps create high-quality, immersive experiences. By outlining a way to represent objects and environments, the standard will help our team make the dynamic raytracer that acts as one of the center points of our project. Following this standard will ensure our applications work well with current and upcoming software. Using ISO/IEC 23488:2022 will help improve the visual quality of our virtual environments and also help create more exciting and realistic interactions within our virtual environments.

We will remain consistent with the ISO/IEC 23488:2022 standard by incorporating the best practices for rendering using our dynamic raytracer. This will include clearly defining our techniques for object representation and ensuring that all objects are rendered as faithfully to their original forms as possible. We will use the guidelines specified within the standard to help enhance the visual quality of our environments and create realistic interaction between our light source and rendered objects.

3 Project Plan

3.1 Project Management/Tracking Procedures

Our team is using an agile workflow with 2-week sprints. We chose agile because we used progressive elaboration to develop our project plan further as we learned more about the technologies and techniques used in Gaussian Splatting. Working in 2-week sprints allowed us to make the most out of this process.

For communication and scheduling, we used Discord. We tracked tasks using the Gitlab issue board, and using Git and branches allowed us to work collaboratively on the project. Finally, we used our weekly reports to document our progress so our advisor (or anyone else) could follow our work from the outside.

3.2 Task Decomposition

Hybrid Gaussian Raytracer:

- Using PBR techniques and spherical harmonics, generate a render of composite scenes with triangles and Gaussian models.
    - Get Scene Data
        - Get Unity camera parameters
        - Emissive materials support
        - Get triangle mesh objects in the scene
            - Generate Axis-Aligned Bounding Boxes using surface area heuristic.
        - Get Gaussian objects in the scene
            - Parse Gaussian splat ply files
            - Generate Axis-Aligned Bounding Boxes by calculating and then sorting the Morton codes for each Gaussian.
    - Create a rendering pipeline.
        - Create a command buffer containing all necessary rendering commands and insert it in the camera's rendering pipeline stage.
            - Preprocess all path buffer information.
                - Clear the current frame buffer
                - Fill a path ends buffer with all of the paths
                - Clear the paths and continue the buffer
                - Clear the temporary paths and continue the buffer
                - Fill the camera information buffer with the current camera parameters
            - Generate all primary rays for paths in the path ends buffer and place them into the paths continue buffer
            - Loop through all paths in the paths continue buffer, and determine if there is an intersection. Add the intersection point to a distance-sorted list of hits. Loop through this hit list after each insertion until path transmittance reaches a minimum—cull remaining intersections. If the path contains a hit, find hit payload values by weighted average given material's opacity, and add the

        path to the temporary paths continue buffer; otherwise, add the path to the paths end buffer.
    - Loop through the temporary paths, continue buffer, and do shading calculations. Place these paths back into the (non-temporary) paths continue buffer.
    - Repeat the intersection and shading code until paths have reached their bounce limit.
    - Accumulate the current frame buffer
    - Increment the frame index

Machine Learning (ML) Models:

- Data Preparation
    - Gather a dataset of well-known and standardized image sets for novel view synthesis
    - Divide the dataset into training, validation, and testing sets.
- Model Selection
    - Select a suitable ML architecture (regression models using SGD).
    - Define input-output mapping (input: images; output: .ply file for optimized point cloud).
- Training the Model
    - Choose an appropriate loss function (e.g., Mean Absolute Error for property estimation).
    - Set up the training configuration (optimizer, epochs).
    - Train the model using the training dataset and validate using the validation set.
- Model Evaluation
    - Evaluate the trained model on the testing dataset.
    - Calculate performance metrics
    - Review results and refine the model or input features if needed.
- Integration and Deployment
    - Integrate the structure from motion and Gaussian optimizer models into the rendering pipeline.
    - Develop a file structure for the Unity package
    - Optimize the model for speed and efficiency in real-time applications.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Graphics / Raytracer Milestones

- Path tracer with triangle support
    - The program should render a scene or object defined using traditional triangle meshes using a path tracer.
    - Performance should be at least 30 FPS
    - Takes in Unity scene data
- Path tracer with material support
    - The path tracer correctly renders triangles with materials and textures attached instead of simple colors.
    - Performance should be at least 30 FPS

- Path tracer with PBR properties
    - Triangles should be rendered based on their physically based rendering properties.
    - Performance should be at least 30 FPS
- Path tracer with basic Gaussian support
    - The path tracer should now allow rays to intersect with one or multiple 3d Gaussians to determine the color of a pixel.
    - Performance should be at least 30 FPS
    - The program should be able to read Gaussian data from a file.
- Path tracer with Gaussian support
    - The path tracer should change the rendering of Gaussians based on the view direction and their spherical harmonic coefficients.
    - The color of the pixels when intersecting Gaussians should now be responsive to changes in lighting.
    - Performance should be at least 30 FPS.

Machine Learning Milestones

- Data Preparation and Feature Extraction
    - Gather and process input video into image frames for Structure from Motion input
    - Identify and extract the proper albedo, opacity, covariance, and spherical harmonics.
    - Deliverable: A cleaned dataset and a documented feature extraction method.
- Model Development and Training
    - Select the ML architecture and define input-output mappings.
    - Train the model using the training dataset and validate it with the validation set.
    - Deliverable: A trained model(s) with validation results.
- Evaluation and Integration
    - Evaluate the model on the testing dataset, refine it as needed, and integrate it into the rendering pipeline.
    - Develop an interface for inputting images and outputting .ply files with optimized Gaussian models.
    - Deliverable: A functioning model integrated into the pipeline, performance metrics, and user interface.

3.4 Project Timeline/Schedule

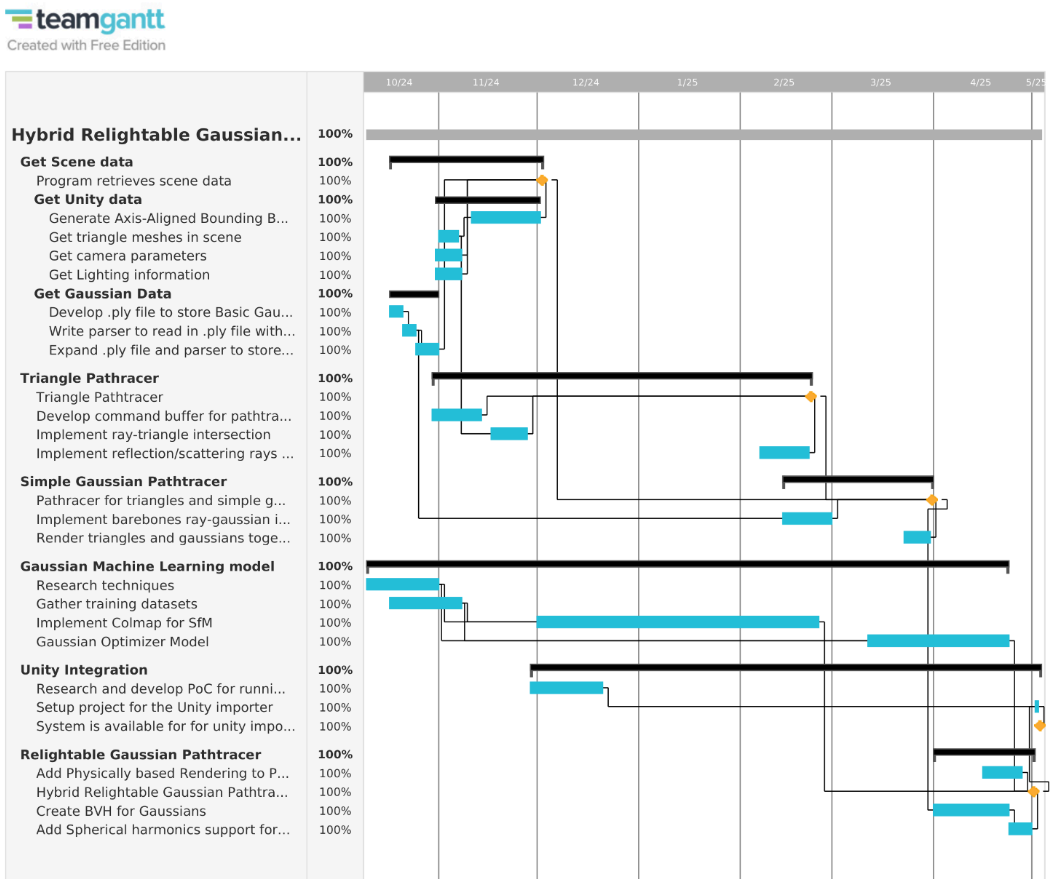


Figure 1: The Gantt chart for our project

In summary, our project was broken into four different deliverables.

Our first deliverable was two demos we presented at our faculty review panel. One was the first implementation of the path tracer, which supported triangles and could render them. This deliverable encompassed all of the tasks in the **Get Unity Data** task group except for Generate Axis Aligned Bounding Boxes, the Develop Command Buffer for pathtracer task, and the implement ray-triangle implementation task. The other demo was a program that rendered 2D cross sections of Gaussians.

The next deliverable was the full triangle path tracer with support for simple Gaussians. This program rendered triangles efficiently using a bounding volume hierarchy with texture support and supported both diffuse and emissive materials, rendered alongside a small number of simple Gaussians. This deliverable encompassed all of the tasks in the Simple Gaussian Pathtracer group, the generate Axis Aligned Bounding Boxes with the surface area heuristic task, and the Implement reflection/scattering rays for triangles task.

The third deliverable is the full Hybrid Relightable Gaussian Pathtracer. In this deliverable, the Pathtracer now renders scenes with triangles and Gaussians. Triangles are rendered using Physically Based Rendering. The graphics side is also now integrated with the machine learning outputs and is able to render Gaussian models created by our machine learning models. This deliverable encompasses all of the Gaussian Machine Learning Model task group and all of the Relightable Gaussian Pathtracer task group.

The final deliverable is to make our project available to others by matching the repository format to the one expected by the Unity importer so that our project is available for others to use.

Milestone	**Completion Date**
The program retrieves scene data from the Unity scene and Gaussian files	11/15/24
Simple Pathtracer for triangles	11/27/24
Demo Gaussian Rendering	11/4/24
Structure from motion	2/24/25
Pathtracer for triangles and small numbers of simple Gaussians	3/31/25
Pathtracer for triangles and Gaussians with Physically-based Rendering for triangles.	4/30/25
Gaussian optimizer model	4/23/25
The project is available for Unity importing	5/2/25

Figure 2: Milestones for the extent of our senior design project.

3.5 Risks and Risk Management/Mitigation

When developing a complex system that combines a Hybrid Gaussian Raytracer with machine learning models for PBR property extraction, several risks can arise throughout the project lifecycle. Identifying and managing these risks is crucial for ensuring successful outcomes and minimizing disruptions.

One of the primary risks is the technical complexity of integrating the raytracing system with the machine learning model. Differences between the data formats used in the rendering pipeline and those required by the ML model could lead to compatibility issues. Early prototypes should be developed to test the data interchange processes to mitigate this risk. Establishing clear interfaces and using standardized formats can advocate for smoother integration. Continuous communication between the teams working on the rendering engine and the machine learning components will also help identify potential issues early.

The success of the machine learning model heavily relies on the quality and representativeness of the training dataset. Risks include insufficient diversity in the point clouds, noise, and outliers that could skew the model's performance. A rigorous data collection and preprocessing pipeline should be implemented to manage this risk, including outlier detection and handling techniques. Regular dataset validation against real-world scenarios will ensure it remains relevant and robust.

Finally, once integrated, the system must be optimized for performance in real-time applications. Risks in this phase include performance degradation due to inefficient algorithms or inadequate hardware resources. Regular profiling of the rendering pipeline and machine learning components can help identify performance bottlenecks early. Additionally, incorporating optimization techniques, such as reducing the complexity of algorithms and leveraging parallel processing, will enhance the system's efficiency.

Our first step to mitigate the risks associated with integrating the two components of the system was to make sure the graphics and machine learning teams were in communication. Later in the project, we encountered an issue where the graphics team had adopted a different method of storing Gaussians that the machine learning models output. This was solved by changing the parser used by the graphics program to read in Gaussians to match the machine learning models' output.

To overcome the risk of bad data sets, we made sure that the datasets we used were well-known pre-existing data sets that were known to be accurate and good for the purposes we wanted them for.

In order to mitigate the risks of poor performance, we have used several performance tests to ensure that our performance improvement methods, such as the use of bounding volume hierarchies, have been effective. Notably, during the code review process for the implementation of the BVH, we compared the performance of the same scene on the branch containing the BVH and one without; this test demonstrated a substantial performance increase when using the BVH.

3.6 Personnel Effort Requirements

Predicted Task Time:

Task	**Person-Hours**
Generate Axis-Aligned Bounding Boxes using surface area heuristic.	6
Get triangle meshes in the scene	4
Get camera parameters	2

Get Lighting information	2
Develop a .ply file to store Basic Gaussians	6
Write a parser to read in a .ply file with Gaussians	3
Expand .ply file and parser to store Gaussians	4
Develop a command buffer for the path tracer	10
Implement ray-triangle intersection	8
Implement the ray-basic Gaussian intersection	10
Train the model to determine normal vectors from Basic Gaussians	24
Research machine learning techniques	20
Gather training datasets	10
Train the model to determine PBR properties for Gaussians	36
Implement real-time camera movement	10
Implement reflection/scattering rays for triangles	16
Implement reflection/scattering rays for Gaussians	16
Implement real-time lighting manipulation	10
Research and develop a PoC for running Pytorch inside Unity	20
Create a Unity package for the system	10
Add Physically-based Rendering to Pathtracer	24
Apply PBR to triangles	12
Apply PBR to Gaussians	12
Refine Pathtracer and model	24

Figure 3: Table of predicted lengths of tasks

Actual Task Time:

Task	**Person-Hours**

Generate Axis-Aligned Bounding Boxes using surface area heuristic.	6
Get triangle meshes in the scene	4
Get camera parameters	2
Get Lighting information	2
Develop a .ply file to store Basic Gaussians	6
Write a parser to read in a .ply file with Gaussians	3
Expand .ply file and parser to store Gaussians	4
Develop a command buffer for the path tracer	10
Implement ray-triangle intersection	8
Implement the ray-basic Gaussian intersection	10
Train the model to determine normal vectors from Basic Gaussians	24
Research machine learning techniques	20
Gather training datasets	10
Train the model to determine PBR properties for Gaussians	36
Implement real-time camera movement	10
Implement reflection/scattering rays for triangles	16
Implement reflection/scattering rays for Gaussians	16
Implement real-time lighting manipulation	10
Research and develop a PoC for running PyTorch inside Unity	20
Create a Unity package for the system	10
Add Physically-based Rendering to Path tracer	24
Apply PBR to triangles	12
Apply PBR to Gaussians	12
Refine Pathtracer and model	24

Figure 4: Table with actual task lengths

3.7 Other Resource Requirements

It is crucial to secure adequate computational power to successfully develop and deploy our hybrid Gaussian raytracer and machine learning models. The rendering engine and machine learning components are resource-intensive, requiring significant processing capabilities to efficiently handle complex calculations and large datasets. The rendering engine supports real-time performance and processing high-quality graphics, which demands powerful GPUs and ample memory.

Simultaneously, machine learning models require robust computational resources for training and inference, mainly when dealing with extensive datasets of point clouds. To mitigate potential performance bottlenecks, we investigated high-performance computing infrastructure that seamlessly supports both the rendering tasks and the machine learning processes, ensuring that the overall system operates efficiently and meets project timelines.

Given the nature of our project, an average desktop computer meets our computational needs. The hybrid Gaussian raytracer and the machine learning models can be efficiently run on standard hardware to optimize resource usage. While our system processes point clouds and renders complex scenes, the operations involved are manageable within the capabilities of a typical consumer-grade machine. Therefore, we don't require high-end computing resources, making the project accessible and cost-effective. This allowed us to focus on development without needing specialized or expensive hardware.

4 Design

4.1 Design Context

4.1.1 Broader Context

Our project impacts the art sector of the public sphere, as our project generates realistic 3D models from videos. In particular, the communities affected are 3D modelers who create physically based models and textures, as well as Unity users who would be interested in creating their own 3D Gaussian models.

Area	**Description**	**Examples**
Public health, safety, and welfare	It should require the user to be behind the screen less.	It should increase the speed and quality of creating custom 3D objects.
Global, cultural, and social	This project reflects the hard work and the magic of working together as a team.	The use of this product should elate the user and inspire them to create products that can help others.
Environmental	Increased electrical usage	ML and Ray Tracing are computationally expensive. This means higher wattage usage.

Economic	Increased output in the custom 3D modeling market. Potentially helping fuel online 3D printing companies.	The user could desire to scan themselves and then print a full-scale version to help make custom props.

Figure 5: Context table for our system

4.1.2 Prior Work/Solutions

There are other similar Gaussian ray tracer products out there. They do the same steps of generating a 3D Gaussian model using Structure-from-Motion and Stochastic Gradient Descent. They build their own render pipeline that either uses rasterization or point-based ray tracing to render these 3D Gaussian models. What makes ours unique is that it incorporates Unity to create a hybrid scene of Gaussian and triangle meshes. This allows for unique and creative scenes. We generate Gaussians in the same way, but we then use Unity as a base to author scenes and game engine logic. Our hybrid renderer overrides the basic Unity rendering pipeline and is capable of rendering both triangle meshes and Gaussian models. Unity is capable of handling a diverse set of 3D file formats, which allows easy drag and drop of traditional polygon models.

	Our Project	**Relightable Gaussian Ray Tracing**
Pros	- Generates 3D Gaussian models from video - Renders both Gaussians and 3D meshes - Unity Compatible	- Generates 3D Gaussian models from video - Relightable Gaussians
Cons	- Gaussian optimization can only run on NVIDIA graphics cards - The quality of the Gaussian model is lower	- Requires NVIDIA graphics cards - Does not implement triangle meshes - It's not Unity compatible

Figure 6: Pro and cons table for our system and alternatives

B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," ACM Transactions on Graphics, vol. 42, no. 4, Jul. 4AD, [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Gao, J., et al. "Relightable 3D Gaussian: Real-time Point Cloud Relighting with BRDF Decomposition and Ray Tracing," in arXiv:2311.16043, 2023.

Nicolas Moenne-Loccoz, undefined., et al. "3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes," in ACM Transactions on Graphics and SIGGRAPH Asia, 2024.

4.1.3 Technical Complexity

Our design consists of 13 integrated components, illustrated in Figure 8 (section 4.3.2). The core elements of the project are a Gaussian ray tracer and a PyTorch-based optimization model, both of which are utilized from within the Unity environment. This integration allows us to take advantage of Unity's rendering pipeline and user interface, making the system accessible to developers familiar with Unity.

The ray tracer component requires a deep understanding of modern graphics rendering techniques, including linear algebra, light interaction, and 3D worldspace design. Specifically, the Gaussian-based rendering system involves handling a collection of 3D Gaussian primitives and modeling how their spatial and visual properties contribute to the final image. This is a technically demanding task that diverges from traditional rasterization and ray tracing based on triangle meshes.

The PyTorch optimizer is responsible for refining the parameters of the Gaussian point cloud. These parameters, such as position, opacity, covariance, and spherical harmonics, are optimized based on a comparison between rendered images and ground-truth input images. This process is repeated each time a new video or image set is provided, enabling dynamic scene reconstruction and optimization.

Our system introduces a hybrid rendering approach by combining Gaussian splats with traditional triangle meshes, which adds a layer of complexity and distinguishes our method from current standards in real-time rendering. The project's performance targets included achieving a runtime frame rate of 30 FPS and completing the full optimization pipeline in under one hour, making the system both visually performant and time-efficient.

4.2 Design Exploration

4.2.1 Design Decisions

We chose to use the Unity engine as the framework for our system. This reduced game engine logic overhead and build complexity for us in running compute shaders and provided a clear path for the packaging and distribution of our system. Using Unity also provides a sense of familiarity for our users as it is one of the largest game engine platforms in the world.

We chose to use a ray tracer in place of Gaussian Rasterization. Some implementations of 3D Gaussian Splatting render their models using rasterization; however, rasterization necessitates ad hoc and inaccurate lighting effect implementations. Because of this, we decided to use a ray tracer, which allows us to have physically accurate control over the lighting and more easily render a hybrid scene containing multiple primitive types.

We chose to target 30 frames per second (FPS). We made this decision because 30 FPS is what is considered to be “real-time” performance.

We did choose to de-scope our project due to time constraints. The feature we decided to remove from our project was the material prediction properties from the optimizer.

4.2.2 Ideation

One of the first choices we made was to use Unity Engine. To make this decision, we discussed alternatives such as using C++ and OpenGL or Vulkan to dispatch our compute shaders. These options require a more complex build pipeline and would make much of our implementation platform dependent. Using the Unity engine abstracts many issues and frees us to focus on other system elements.

4.2.3 Decision-Making and Trade-Off

	Pros	**Cons**
Base OpenGL / Vulkan	No existing render pipeline	It is complicated to build and display output. Different platforms will have different compatibility
Unity Engine	Integration with an already existing tool Built-in systems for dispatching Compute Shaders Unity engine provides solutions for cross-platform compatibility	Our system ignores the existing render pipeline

Figure 7: Pros and cons table for using Base OpenGL/Vulkan and the Unity Engine.

We decided to create our project within the Unity engine; this gave us the benefit of not having to set up our window to display the output, implement our update loop, or handle cross-platform building for low-level code. This decision saved us much time and effort on problems outside our project's core purpose.

The downside of this decision is that Unity has a built-in render pipeline, which our solution does not use, so there is a performance cost to using Unity, but we decided the benefits outweigh the cost.

4.3 Final Design

4.3.1 Overview

The Hybrid Relightable 3D Gaussian Rendering System is a real-time novel view synthesis tool that leverages advanced machine learning techniques, 3D geometry processing, and optimized rendering methods to create high-quality, interactive 3D scenes. The system is designed to work with image-based data (such as video or image sequences) and polygonal models, seamlessly integrating them into a unified rendering pipeline.

The process begins by capturing a video input and extracting frames to splice into a sequence of images, which are then processed using Structure from Motion (SfM) techniques. This process generates point clouds—collections of 3D points representing the scene's structure. From these point clouds, the system creates Gaussian splats, simplified representations of the scene's geometry, with each point in the cloud representing a Gaussian distribution in 3D space.

Next, the generated Gaussian points are fed into a PyTorch-based optimizer. This optimizer utilizes machine learning techniques to refine the positions and properties of the points. Additionally, it creates extra attributes crucial for accurate rendering and details. This optimization ensures the final 3D scene is visually realistic and computationally efficient.

The output from the optimizer and its model is then passed through a parser that prepares the data for rendering. This includes creating a Bounding Volume Hierarchy (BVH), a hierarchical structure that accelerates the ray tracing process by organizing the 3D objects in the scene for more efficient intersection tests. For scenes requiring more detailed geometry, users can integrate polygonal meshes (triangle meshes), which are also processed into their own BVH structures, enabling them to coexist and be rendered alongside the Gaussian splats.

The core of the system's rendering capability lies in its ray tracer. This component takes the BVH data (whether Gaussian or triangle mesh-based). It combines it with the Unity camera to calculate how light interacts within the scene, ultimately producing realistic images. At path intersection points, each hit is inserted into a sorted list of intersections. Post insertion, we loop through the sorted hits list, keeping track of the paths' transmittance as it passes through translucent materials. Once the path's transmittance value reaches a minimum value, we cull the remaining list and consider that the point of intersection. The ray tracing process accounts for various optical effects, such as shadows, reflections, and global illumination, ensuring the final output is both physically accurate and visually compelling.

4.3.2 Detailed Design and Visual(s)

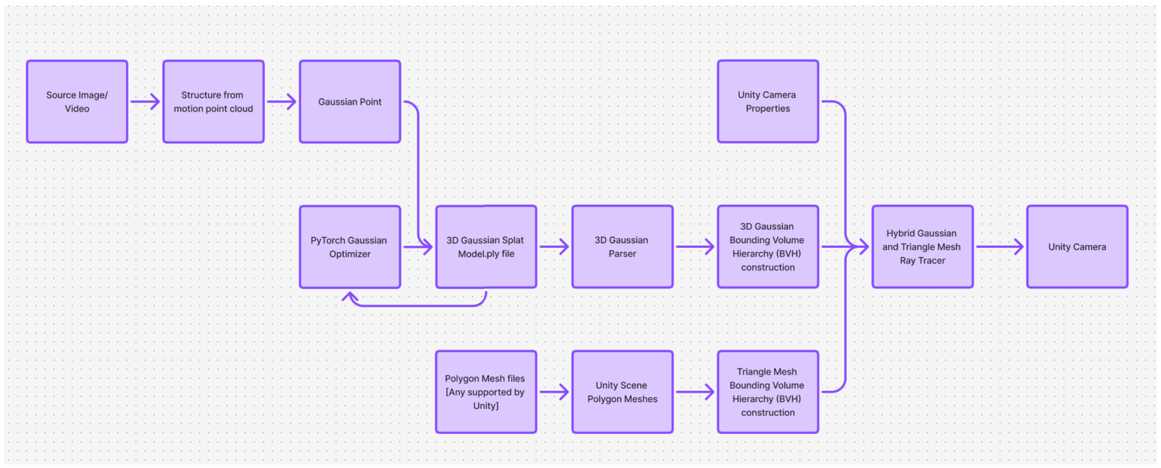

Source Image/ Video
Structure from motion point cloud
Gaussian Point
Unity Camera Properties
PyTorch Gaussian Optimizer
3D Gaussian Splat Model.ply file
3D Gaussian Parser
3D Gaussian Bounding Volume Hierarchy (BVH) construction
Hybrid Gaussian and Triangle Mesh Ray Tracer
Unity Camera
Polygon Mesh files [Any supported by Unity]
Unity Scene Polygon Meshes
Triangle Mesh Bounding Volume Hierarchy (BVH) construction

Figure 8: High-level overview of the Hybrid Relightable 3D Gaussian Rendering System.

As the overview explains, our solution consists of a pipeline that takes in images or videos and ultimately outputs a rendered scene. Our system consists of multiple components along this pipeline that modify the data.

1. The first component of the solution is a structure from the motion solution, which creates a cloud of Gaussians from a series of images or images extracted from videos.
    a. This solution was implemented and requires the user to upload their video to a specific directory, from there the program uses it to create a ply file of a point cloud.
2. The second component is a machine learning model that uses the point cloud as input and converts the point cloud into a Gaussian model.
    a. These Gaussians are then outputted to another ply file so they can be read in and used by the ray tracer.

3. The final component is the raytracer itself. The raytracer should take in the Gaussian output by the machine learning model and any traditional polygon meshes and render them together in a 3D scene.
    a. The raytracer should allow the user to configure the camera's position and lighting by editing the Unity scene or other methods.
    b. For performance reasons, the raytracer should construct a bounding volume hierarchy to accelerate calculating the ray intersections.
    c. The output of the raytracer should be in real-time (30 FPS) and allow the user to update the lighting and camera positions while running.

4.3.3 Functionality

The System is designed to operate seamlessly within a user's workflow, enabling real-time 3D scene reconstruction and rendering. To start, a user will interact with the system through a Unity package, which serves as the interface for input and output. Depending on their needs, the user can upload a video/video-derived image set or a polygon mesh into the Unity package.

Suppose the input is a video or a set of images. In that case, the system will first extract the 3D geometry using Structure from Motion (SfM), then generate a point cloud for further optimization. This point cloud will then be processed through a PyTorch-based optimizer, which refines the points and transforms them into Gaussians. The system then constructs a Bounding Volume Hierarchy (BVH) to organize the scene for efficient ray tracing and rendering. This optimized scene, which includes the optimized Gaussian model, is ready to be rendered with our custom Unity Render Pipeline, ensuring that the lighting and materials are accurately simulated in real time.

If the user uploads a polygon mesh, the system similarly processes the geometry through BVH construction without the SfM step. The polygon mesh will be integrated into the scene alongside the 3D Gaussian splats, with ray tracing performed on both geometry types for optimal performance.

Once the scene is rendered, the user can interact with the final output directly within Unity or export the rendered scene for use in various applications. The system provides flexibility, allowing users to apply the rendered 3D scenes to their specific use cases, whether for simulations, visualizations, or immersive experiences.

4.3.4 Areas of Challenge

The ML problems we ran into did take longer to solve than originally hoped for, which caused time setbacks that caused us to descope some additional Gaussian parameters, including material prediction and normal direction. There were the everyday issues of communication and finding time to meet with one another to ensure constant, consistent progress in the project.

We solved these problems by discussing them as a group and trying to solve them. From there, we learned what we didn't know to create new-to-us solutions that fit our problems. We were transparent in communicating with each other about issues we didn't know how to solve, or giving notice when we wouldn't be able to come to the meeting.

4.4 Technology Considerations

A major trade-off of our project is that the user must have an NVIDIA graphics card to run our program. This requirement limits who can use our program, but the trade-off is that the custom 3D models should be of higher quality due to the ML optimizer that is generating more data from the image than simply just the Gaussian points. This should be an overall strength to our project, as it is likely that whoever is interested in using our program would already meet this requirement. Another major trade-off is the design choice to make this a Unity project vs other options. This should again be a strength of the project, as Unity is a very common game engine used by many people.

5 Testing

Our testing approach prioritized early and iterative integration testing to ensure all system components aligned with the design requirements. Due to the modular structure of our project, particular attention was given to maintaining consistency in inputs and outputs across modules. This strategy helped ensure seamless integration between subsystems. The following sections describe the methods and tools we used to achieve this.

5.1 Unit Testing

Our machine learning unit tests involve comparing a baseline image with a newly generated image from our code. We perform a pixel-by-pixel comparison, allowing for a small margin of error. To facilitate testing, we developed custom visualizers to compare and inspect the images more effectively. These visual tools not only streamlined the debugging process but also helped us quickly identify discrepancies and verify the correctness of our image generation pipeline. This approach ensured the robustness and consistency of our machine learning outputs throughout development.

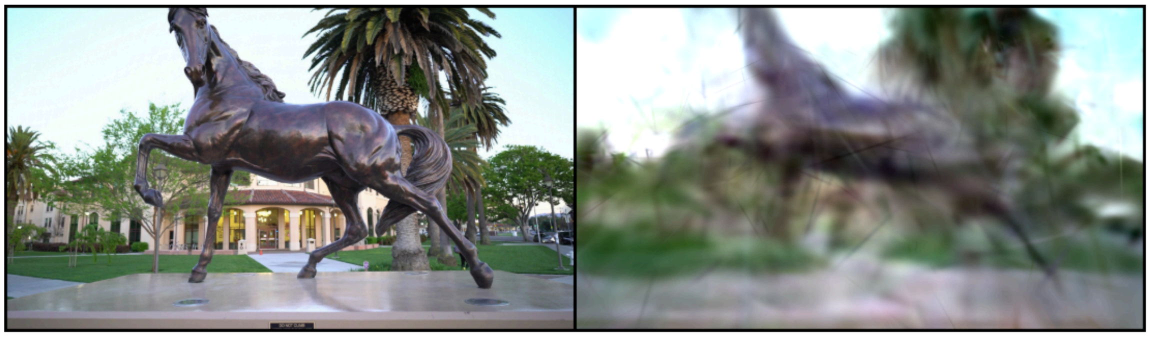


Figure 9: Comparison of the pre-optimized truth image (left) and the optimized Gaussian model using the optimizer's renderer(right) from a horse image set.

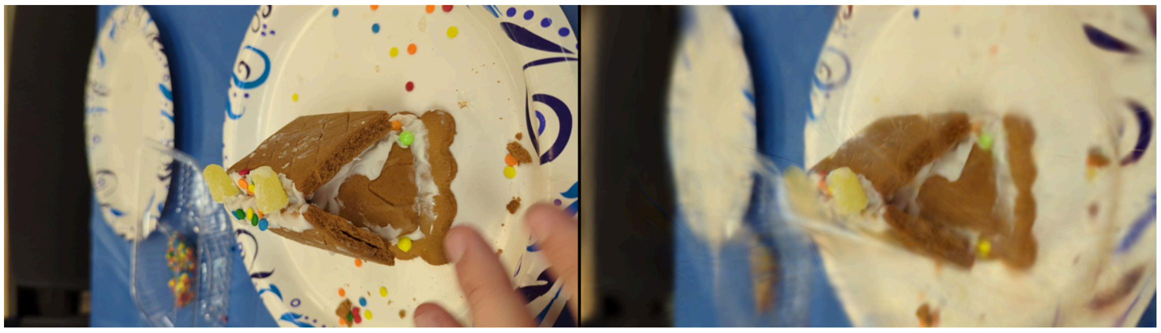


Figure 10: Comparison of the pre-optimized truth image (left) extracted from a video and the optimized Gaussian model using the optimizer's renderer (right) for a gingerbread house.

For graphics unit testing, the team initiated testing after the completion of the Gaussian Unity parser. Initial efforts focused on rendering a single Gaussian to validate the core rendering pipeline and ensure accurate visual representation. Once this baseline functionality was confirmed, we progressed to more complex scenes involving multiple Gaussians and their intersections. These tests allowed us to verify spatial accuracy, depth handling, and visual consistency within the Unity rendering engine. This step-by-step approach ensured a reliable foundation before integrating more advanced graphical features.

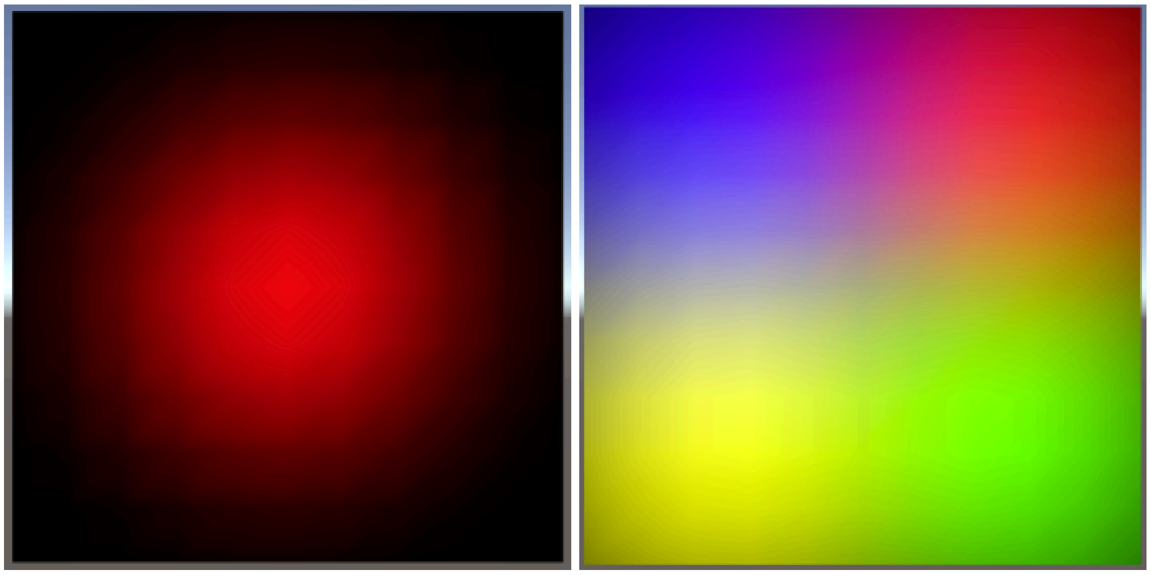


Figure 11: Comparison of a single Gaussian scene(left) and a simple Gaussian intersection scene (right) from testing.

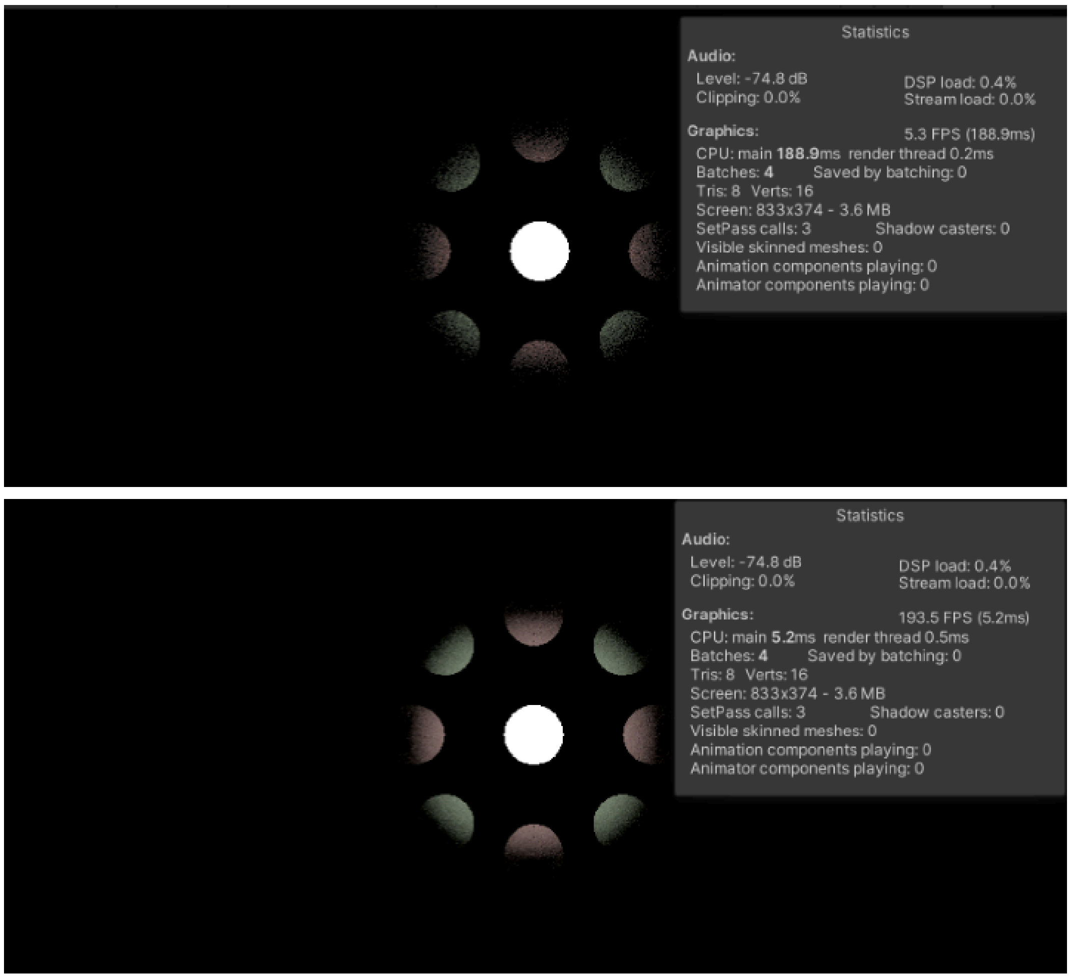
5.2 Interface Testing

Our project has relatively few custom interfaces due to its deep integration with Unity. As a result, we focused on leveraging Unity's built-in interfaces to communicate program status. Specifically, we utilized Unity's output logs to provide users with updates on the program's progress. This approach allowed us to maintain a streamlined design while still offering transparency and useful feedback during execution.

5.3 Integration Testing

The modular nature of our project necessitated a strong emphasis on integration between components. Once the initial development of each module was complete, we created a control script to manage the flow of inputs, outputs, and the sequential execution of these modules. Our integration testing centered around the PythonScriptRunner.py, which orchestrates the entire pipeline: it first executes the video-to-image script, then selects the appropriate output directory and runs the Structure-from-Motion (SfM) script to generate a point cloud. This point cloud is then passed into the optimizer, which produces the final model for rendering.

This integration testing process was critical in identifying discrepancies between input and output paths, as well as inconsistencies in data formatting. It played a key role in ensuring seamless coordination between modules and significantly streamlined the development process.



```
Statistics
Audio:
  Level: -74.8 dB                DSP load: 0.4%
  Clipping: 0.0%                 Stream load: 0.0%
Graphics:                        5.3 FPS (188.9ms)
  CPU: main 188.9ms  render thread 0.2ms
  Batches: 4       Saved by batching: 0
  Tris: 8  Verts: 16
  Screen: 833x374 - 3.6 MB
  SetPass calls: 3               Shadow casters: 0
  Visible skinned meshes: 0
  Animation components playing: 0
  Animator components playing: 0
```

```
Statistics
Audio:
  Level: -74.8 dB                DSP load: 0.4%
  Clipping: 0.0%                 Stream load: 0.0%
Graphics:                        193.5 FPS (5.2ms)
  CPU: main 5.2ms  render thread 0.5ms
  Batches: 4       Saved by batching: 0
  Tris: 8  Verts: 16
  Screen: 833x374 - 3.6 MB
  SetPass calls: 3               Shadow casters: 0
  Visible skinned meshes: 0
  Animation components playing: 0
  Animator components playing: 0
```

Figure 12: Performance comparison, in FPS, between a scene without a BVH (top) and one with a BVH (bottom).

5.4 System Testing

System-level testing in our project combines unit, interface, and integration tests to validate the end-to-end functionality of the system. A typical test involves rendering a complex scene composed of a Gaussian splat model, which is generated from Structure-from-Motion (SfM) and machine learning models, as well as traditional polygon meshes. The final rendered output is verified for both visual accuracy and performance. Key requirements tested include rendering speed and image quality. We used Unity Editor's Play Mode Tests along with performance benchmarking tools to automate and streamline this process.

Although the project is modular in design, it follows a mostly linear processing structure. This design is beneficial because it allows the workflow to begin with a video or image set as input. Once Play Mode is entered in Unity, the program automatically processes the input through each module. After processing, the final scene is rendered in Unity, where it can be observed and evaluated for output quality.

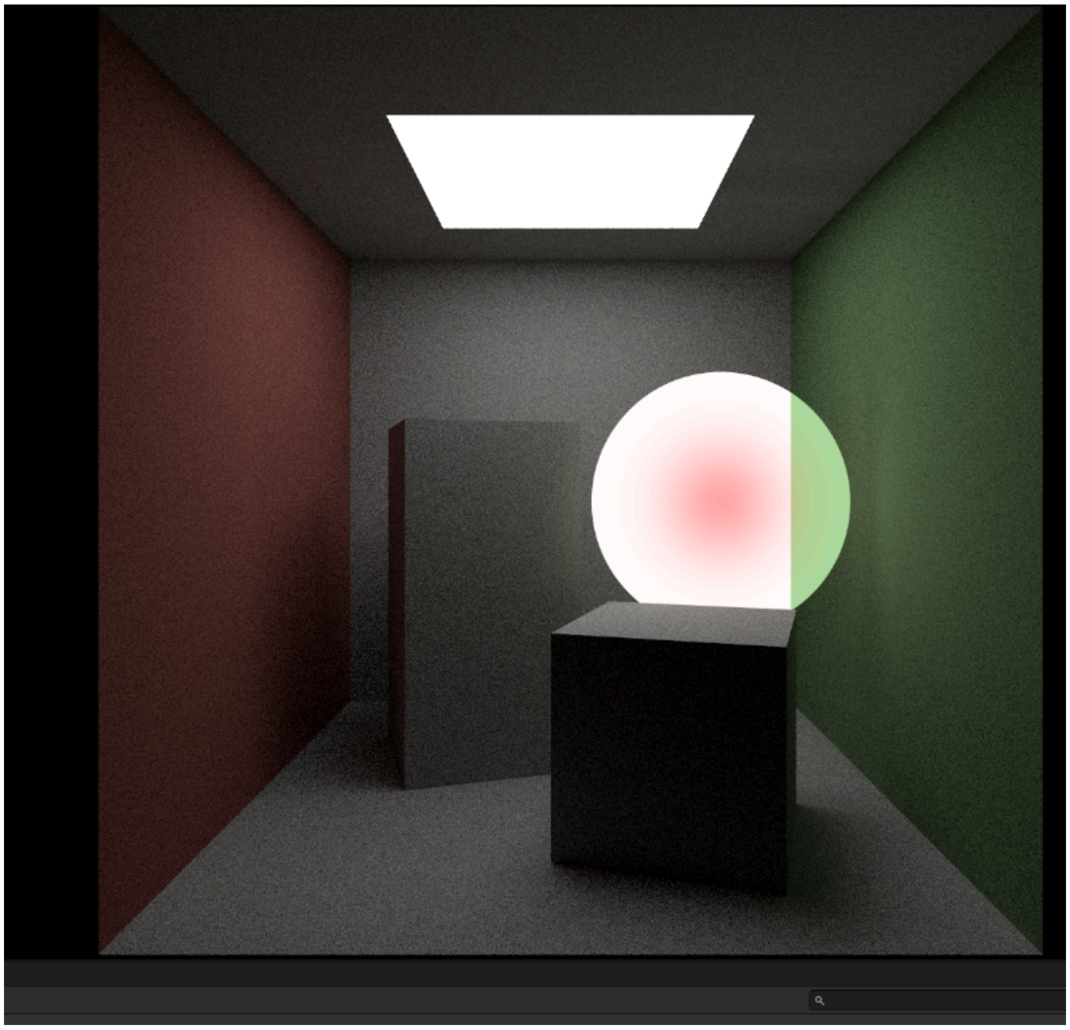


Figure 13: Cornel Box with a single Gaussian

5.5 Regression Testing

Our project does not require a traditional regression testing suite, as the package operates relatively independently from the Unity environment. The machine learning component mainly consists of standalone scripts linked together through a script runner. As such, the primary focus of regression testing is to ensure that the Python script runner remains compatible with new Unity scenes. Additionally, we need to verify that changes to Unity do not negatively affect the renderer, which, given the design, should not be an issue.

5.6 Acceptance Testing

We ensure that design requirements are validated through continuous communication with the machine learning team, the graphics team, and the professor overseeing our work. Demonstrations involved running test scenes with predefined criteria, such as a basic Gaussian intersection scene.

As this is a student-proposed project, the acceptance testing was conducted by us, the developers. While this was advantageous in that we had a clear understanding of what was possible and what the final product should look like, it also presented a challenge. Our deep familiarity with the project made it difficult to assess the work objectively. Given more time, we would have benefited from testing with impartial users to gain more objective feedback on the final product.

5.7 User Testing

Testing user needs for our project is straightforward, as the primary objectives were performance-based. For instance, we aimed to process the Structure-from-Motion (SfM) and optimization tasks in under one hour. Testing this requirement is simple: we run the program, and if it completes within the specified time frame, the test passes. If the processing time exceeds the target, we revisit our performance optimization techniques.

Similarly, for graphics testing, performance is verified by monitoring the frame rate during the rendering of the Gaussian models. Our goal was to achieve a minimum of 30 frames per second, which has been successfully met on most systems. The frame rate typically ranges from 30 to 200 frames per second, depending on the hardware specifications.

5.8 Compatibility Testing

Since our project involves optimization tasks of non-trivial size, we leveraged CUDA support where possible to accelerate performance. Many components are compatible with both GPU and CPU processing, but by default, we prioritize GPU execution using CUDA for optimal speed. In cases where GPU memory is insufficient, particularly during the Structure-from-Motion (SfM) stage, we automatically fall back to multithreaded CPU processing to ensure continued functionality and stability.

5.9 Results

Our testing efforts played a critical role in both development and validation. Integration tests were especially helpful in identifying mismatches between module inputs and outputs early in the development cycle, allowing us to address issues before full system assembly. In parallel, performance testing provided valuable insights into how well the system met our efficiency goals.

Given the computationally intensive and graphics-heavy nature of our project, performance was a primary concern. We prioritized smooth rendering and reasonable processing times to ensure usability. Our results indicate that the system performs in line with our expectations.

We successfully achieved our goal of a minimum of 30 frames per second during Gaussian model rendering, even on mid-range hardware. On high-end systems, frame rates typically ranged between 30 and 200 frames per second, depending on model complexity and scene content. Processing time for the full pipeline—including video-to-image conversion, Structure-from-Motion, and optimization—averaged around one hour on our best available hardware. This aligns with our target of completing processing within an hour for average scenes. While more complex models or less capable hardware resulted in slightly longer runtimes, the system consistently stayed within an acceptable range.

Overall, our results validate that the project meets its primary requirements for performance and responsiveness. With further optimization and hardware tuning, even greater efficiency could be achieved.

6 Implementation

Our team's development was split into two primary groups: the computer graphics and Machine Learning groups. This was done to efficiently develop the independent core components of our system in parallel. The graphics team is responsible for developing the hybrid ray tracing rendering solution in the Unity game engine, and the ML team is tasked with generating a point cloud given a series of real-world input images or video using Structure from Motion (SfM), which will then be given to a machine learning model to create and optimize a Gaussian point cloud.

During the first semester, the graphics team primarily focused on creating the foundations for our hybrid rendering solution. This consisted of overriding Unity’s rendering pipeline with our own custom one. The Unity scripting API allows you to define new command buffers, a series of commands you wish the GPU to execute, during a specific Unity rendering loop phase. The general architecture of our hybrid renderer inserts itself at the end of everything, \`CameraEvent.AfterEverything\`. There, it overrides the image Unity’s universal rendering pipeline generated with a new ray-traced image. To create the ray-traced image, a ray is generated for each pixel, originating from the camera’s origin and passing through the center of the pixel. For each generated ray, we determine if it collides with anything in the scene. If a ray collides with an object, it performs a series of lighting calculations depending on material properties, incident direction, and incoming light direction. After this lighting calculation, we reflect this ray off the object’s surface and continue tracing it through the scene. Once all rays have reached some arbitrary bounce limit, the image has finished rendering.

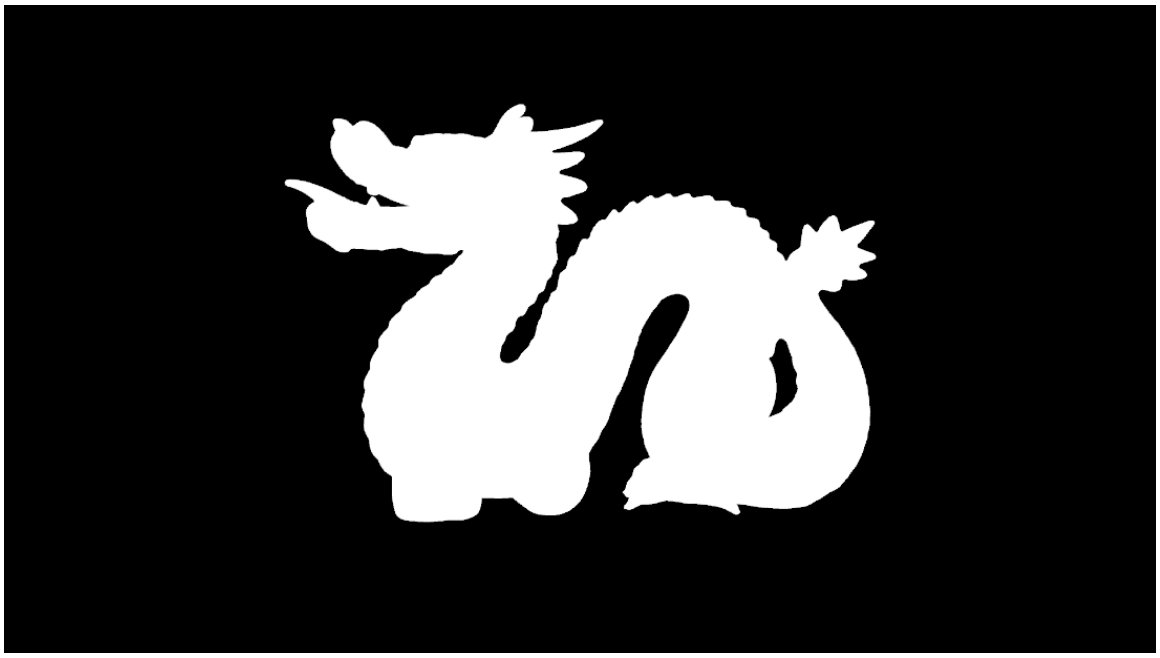


Figure 14: A ray-traced image of the Stanford Dragon with no scene lighting and a solid color unlit material.

For simplicity, the graphics team focused primarily on triangle mesh integration before Gaussian model integration due to the abundant resources on the former. This allows us to guarantee the accuracy of our physically based lighting calculations before adding in the additional layer of complexity of non-traditional models. The graphics team's end-of-semester one deliverable consisted of a traditional pinhole camera that can render triangle mesh objects with emissive materials, serves as our scene's light source, and Lambertian diffuse materials, the most basic physically based material.

Even with the parallelized nature of performing these calculations via compute shaders, checking for ray intersections for every triangle in the scene is extremely costly. To sidestep this issue, we use bounding volume hierarchies, an industry-standard solution to this issue, to drastically increase the performance of our ray tracer. A ray will only consider triangles that lie within a volume. Thus, we can create a structure of embedded volumes or bounding boxes, where we recursively check for ray collisions against these bounding boxes.

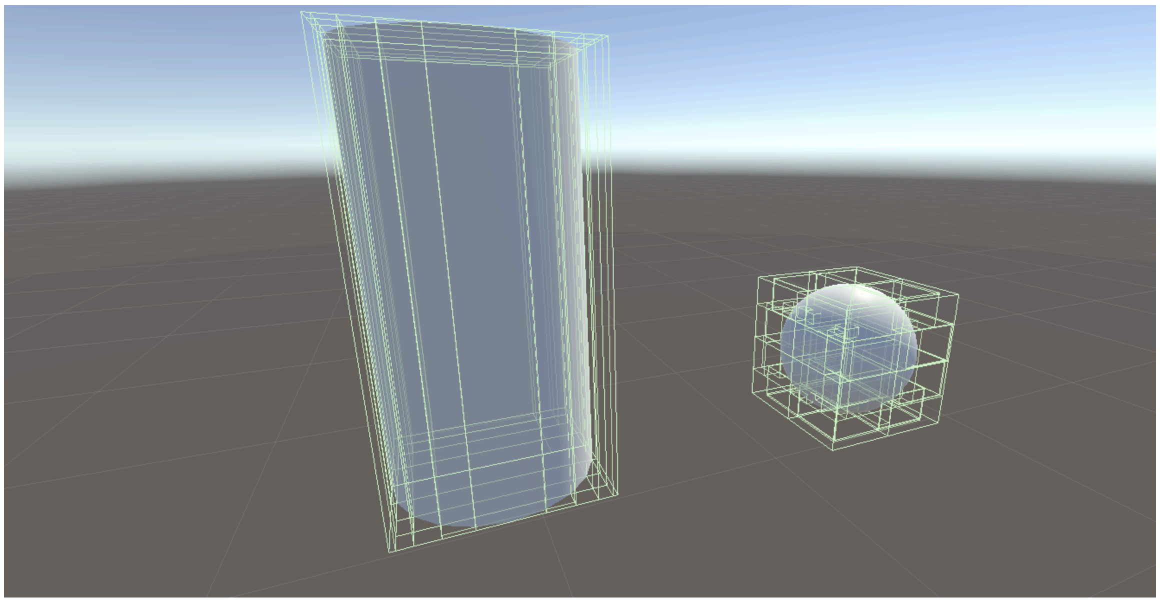


Figure 15: A cylinder and cube with their AABBs visible

This semester, the computer graphics team focused on finishing a diffuse triangle mesh path tracer, physically based materials, and hybrid, multi-intersection Gaussian primitive rendering support. The ray tracer is now capable of determining the reflection direction of the ray at intersection points using cosine-weighted, for diffuse materials, or GGX, for PBR materials, importance sampling. Rendered frames are accumulated in an accumulation buffer, meaning that, over time, the output image will reach its appropriate color using Monte Carlo integration.

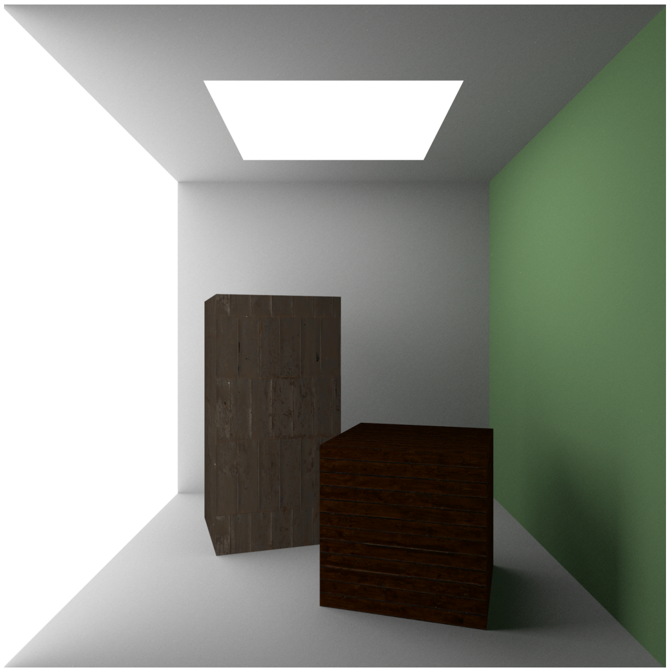


Figure 16: A modified Cornell Box scene with PBR materials.

The other major focus of the graphics team this semester was rendering Gaussians. This was a core component of our project and presented several challenges. The first step we took was to set up a simple ray-gaussian intersection. These scenes had rays intersect with a small number of gaussians; we couldn't do a large number of gaussians yet for performance reasons, and could derive pixel color based on the albedo of the gaussian and the calculated opacity at the point.

In order to improve this simple intersection, we needed to take two steps. The first step was to take in multiple intersections and sort them to calculate the color of a pixel within which the ray passes through many gaussians. The next major component we accomplished was to create a bounding volume hierarchy for Gaussians. Without a BVH, if we had more than a couple of dozen Gaussians, our program would crash, and the models we were hoping to render contained multiple tens of thousands of Gaussians.

To create the BVH for Gaussians, we followed the algorithm used in J. Gao et al, which constructs a BVH in four steps. The first was to calculate the Morton code of every Gaussian. A Morton code is a way of compressing the three-dimensional coordinates into a single number. The next step is to sort the Gaussians in ascending order of the Morton code. The next two steps involved constructing the bounding boxes. First, we built a bounding box for each Gaussian, and then we recursively created bounding boxes that encompassed each two adjacent bounding boxes in the layer below. An important modification we made to the algorithm from the paper was that we implemented it on the CPU instead of the GPU; because of this, we had to multithread the AABB construction in order to ensure it performed well. Once the BVH was implemented, we were able to render full Gaussian models with acceptable performance.

The final element of focus for Gaussians was to implement spherical harmonics. Gaussian models encode color using spherical harmonics because they allow for view-dependent color to be efficiently encoded. We modified our intersection script to calculate the ponderated sum of the spherical harmonics equations for each color channel and used this color in place of the albedo going forward.

Combining the implementation of Gaussians with physically based rendering completed the graphics portion of the system and allowed the system to render high-quality Gaussian and traditional polygon models.

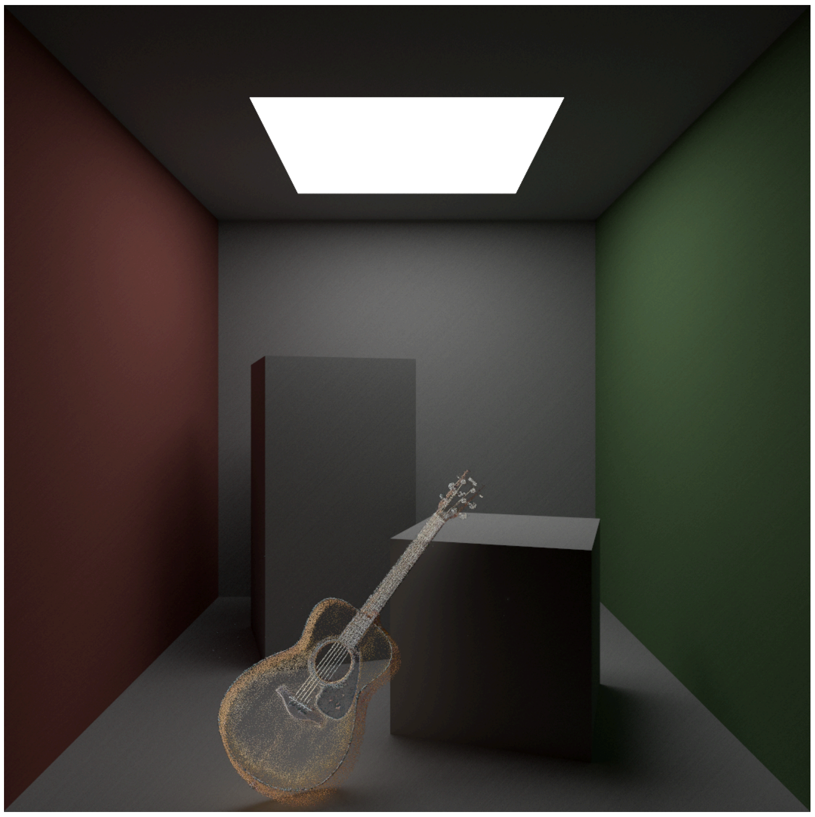


Figure 17: A Gaussian model of a guitar rendered alongside the Cornell box scene

The primary focus of the ML team in the first semester was to generate a 3D point cloud from a sequence of scene images using the Structure-from-Motion (SfM) computer vision method. To accomplish this, we used the open-source program COLMAP along with its Python interface, PyCOLMAP. We tested our dataset with COLMAP to verify that the generated point clouds met our quality requirements and were compatible with the design of our Gaussian Optimizer.

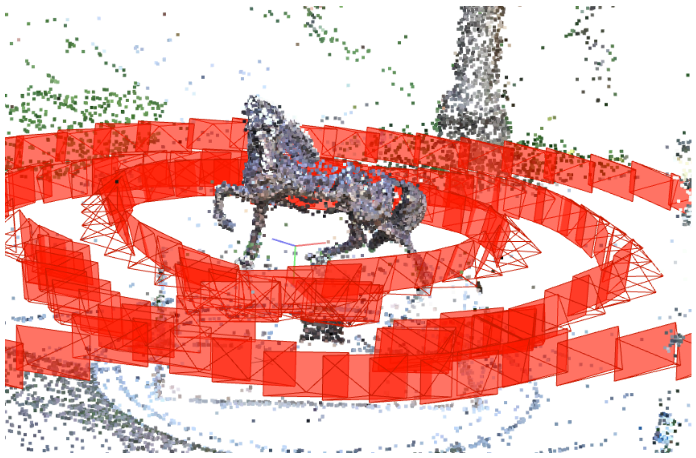


Figure 18: A generated point cloud of a horse statue using COLMAP. The red boxes represent camera/picture locations.

The generated point clouds serve as input to our optimizer model, which converts them into Gaussian point clouds by optimizing parameters such as position, opacity, covariance, and spherical harmonics. The optimization process involves rendering images from the Gaussian representation using the same camera parameters as the original dataset, comparing them to the ground-truth images, and iteratively refining the Gaussian parameters to minimize visual discrepancies. This results in an optimized Gaussian point cloud that closely matches the visual appearance of the original scene.

Our team additionally focused our efforts on ensuring that our Unity package is available to be imported into any standard Unity project. While we were not able to upload it into the asset store due to time constraints and, additionally, due to the inclusion of third-party Python scripts/libraries, we were still able to refine the package to the point where a user can import it using a GitHub link and the Unity package manager. The instructions for importing our package are included in the README for the repository.

6.1 Design Analysis

Our design was ambitious and paved a new feature for Gaussian ray tracers, that being the hybrid element of our project. The optimizer correctly extracts data from video scenes and successfully passes it off to the ray tracer. The ray tracer, in turn, successfully outputs a hybrid scene of triangle meshes and Gaussians rendered inside the Unity pipeline. There are a lot of moving parts to our project, and they all successfully came together to form a functional project.

There are many ways for our project to be improved. These include improving the quality of Gaussian models, the speed of rendering, and implementing a material predictor. We underestimated the time needed to implement the original features we sought to implement and had to de-scope the project due to time constraints. The feature we removed was the relightability of the Gaussians. We unfortunately had to remove this from our scope due to time constraints. Additionally, we wanted to be able to add new lighting in scenes and have it accurately display the new light on the Gaussians. By cutting these features, our package, rather than extracting lighting information for the Gaussian models, simply uses the light that is in the original video used to generate the Gaussian. It is worth noting that triangle meshes are relightable.

7 Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	**Definition**	**ACM Code of Ethics and Project Application**
Work Competence	Provide work demonstrating excellence, high integrity, punctuality, and professional competence.	**2.6 Perform work only in areas of competence.** Our project involves advanced topics such as GPU graphics programming, the Unity engine, and machine learning models using Pytorch. Ensuring work competence, continuous learning, and practice from all team members is ongoing.

Financial Responsibility	Provide products at a reasonable cost in comparison to the quality of the product available.	**1.3 Be honest and trustworthy** Due to our plan to create a Unity package and project circumstances, our product will be free to use.
Communication honesty	Report work understandably and truthfully in such a way as to avoid deception.	**1.3 Be honest and trustworthy** To ensure communication honesty, as a team, we meet each Monday to discuss achievements and create a weekly report that is reviewed by each individual to ensure no deceptions are reported.
Health, Safety, Well-being	Minimize risks to individuals' safety, well-being, and health by adhering to safety standards and actively addressing hazards.	**1.2 Avoid Harm** We plan to integrate industry standards like traditional Polygon meshes into our renderer to avoid harm to potential jobs and job states of current 3D artists.
Property Ownership	Respect the property and ideas of others by recognizing their rights and avoiding unauthorized use or infringement.	**1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.** For our project, we plan to credit all papers, libraries, including the SfM COLMAP library, and additional resources used to create our final project.
Sustainability	Protect the environment both on a local and global scale through ethical practices and by avoiding overconsumption.	**1.2 Avoid Harm** We understand that machine learning models and computer graphics can be computationally expensive and require lots of energy, so to alleviate this issue, we plan to include BVH into our renderer, among other optimization techniques, to limit rigorous computations and energy usage.
Social Responsibility	Create a product that contributes positively to society by addressing key needs, improving quality of life, and promoting overall well-being.	**1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.** Our system will directly benefit our users and stakeholders by offering a faster, higher-quality, more flexible solution to the current industry standard for Novel View Synthesis

Figure 19: Areas of responsibility followed by the corresponding definition, then the ACM Code of Ethics, and the Project Application

Our team is performing exceptionally well in the area of work competence. This is crucial in our project, which involves complex and advanced topics like GPU graphics programming, the Unity engine, and machine learning models using PyTorch. We demonstrate excellence by ensuring every team member continuously learns and hones their skills. This commitment to ongoing education, whether through self-study, peer learning, or practical application, allows us to provide high-quality, professional work. Each member is encouraged to work within their areas of competence, which results in timely, accurate, and impactful contributions to the project. Individual areas of competence were determined during our first few team meetings, personal interests, and previous skills/experience. Our dedication to maintaining high integrity and punctuality in delivering tasks ensures that we consistently meet our objectives and deadlines.

While we perform well in work competence, we recognize the need for improvement in communication and honesty. We hold weekly meetings to discuss our progress and generate a report that each team member reviews to ensure transparency. However, there have been occasions where the clarity and thoroughness of our task documentation could be improved. To fix this, we implemented more rigorous task documentation practices to enhance communication honesty. This helped us ensure that each task was thoroughly documented with clear progress updates, challenges, and solutions. By doing so, we avoided any potential misunderstandings or misrepresentations of our work, ensuring that all reports are truthful and understandable. For example, during the final sprint of our project, we would send daily updates to each other that explained what we accomplished that day, any setbacks or problems we encountered, and how much we were following our preplanned timeline. This improvement helped strengthen our team's ability to report progress transparently and accurately.

Our ethical standards have remained the same since last semester, as the foundation of our project has not altered greatly from our original plan from the previous semester. Since no new functionalities or changes were introduced to our project, our original ACM Code of Ethics continues to be a guide we can follow.

7.2 Four Principles

	Beneficence	**Nonmaleficence**	**Respect for Autonomy**	**Justice**
Public health, safety, and welfare	Our system will directly benefit our users and stakeholders by offering a faster, higher-quality, more flexible solution to the current industry standard for Novel View Synthesis	Our system does directly or indirectly harm our stakeholders and users	After creating a minimum viable product, we plan to send out prototypes to stakeholders and receive user feedback before the system is finalized	Our system will be neatly contained in a Unity Package and will be freely available to download on the Unity Asset Store

Global, cultural, and social	Our system is advancing the research of the computer graphics and machine learning industries, and its mainstream adoption	Our system does not go against or disrespect any particular culture or social group	Our system could support cultural practice by digitally capturing one-to-one recreations of culturally significant locations (e.g., religious sites)	Our system will remain culturally accessible by offering a visual demonstration of how to use our system
Environmental	Our system captures the beauty of real-world environments, which may inspire users to be more environmentally conscious	Even though our system uses machine learning, our model's training is not computationally expensive when compared to LLMs	Our system does not require the user to re-train the Gaussian model locally	Our system will not affect any environmental habitat
Economic	Our system increases the production quality of renders for smaller companies, therefore creating more competition	We mitigate any potential 3D artist job loss by allowing full support of industry-standard tooling	We will make our product free of charge	Our system will not financially affect our users

Figure 20: Four ethics principles chart and how it applies to our project.

The benefit we are working towards is a faster, higher-quality, and more flexible solution for Novel View Synthesis, which will directly enhance user experiences. We aim to deliver faster results without compromising quality by leveraging advanced algorithms and optimizing computational efficiency. Additionally, we incorporated adaptive features that allow users to customize the solution to their specific needs, ensuring greater flexibility. We continuously tested the system with real-world data to ensure success and refine our models based on iterative improvements.

One broader context-principle pair in which our project may be lacking is the potential impact on 3D artists' job security due to automation and advanced tools. While our system supports industry-standard tooling and enhances efficiency, this could inadvertently raise concerns about job displacement in the 3D artist community.

To overcome this, we focus on the positive aspects of the design, which empower 3D artists rather than replace them. By integrating industry-standard tools, including traditional polygon meshes and our Gaussians, we enable artists to leverage automation to reduce repetitive tasks, allowing them to focus on more creative and complex aspects of their work.

7.3 Virtues

7.3.1 Team Virtues

Collaboration is central to our success, as we rely on diverse expertise and input to create an innovative solution. To foster cooperation, we prioritize open communication and a culture of trust where each team member feels comfortable sharing ideas and feedback. We hold weekly team meetings and biweekly meetings with our advisor, Dr Mitra, to discuss progress, challenges, and brainstorming sessions to ensure everyone is aligned on goals. Additionally, we use collaborative tools and platforms, such as Google Drive (docs, slides, etc) and Figma, that allow for real-time sharing and problem-solving, promoting collective ownership of the project.

Innovation is another central virtue of our team. Innovation is the driving force for our approach to solving Novel View Synthesis problems. We encourage innovation by creating a team environment where experimentation is encouraged, and each team member can propose and test new ideas. We also promote innovation by continuing to explore new algorithms, techniques, libraries, and software to continue improving our project's performance and quality. We also prioritize learning and staying up to date with the latest papers and research, with some being released during our design process. All in all, innovation ensures that our team has the knowledge and resources to innovate effectively.

Empathy is an essential part of our virtues, as empathy allows us to understand our users' needs and potential areas of concern. As previously mentioned, we aim to create a Unity package that assists in the creative process for 3D artists by including our ML tools rather than outright replacing the work they are assigned. We plan to continue addressing this virtue by gathering and analyzing user feedback to ensure our solution meets their needs and genuinely assists and benefits their work. We are mindful of how our package will affect the user's workflow and will continue to empathize to ensure the best product we can provide.

7.3.2 Jackson Vanderheyden's Virtues

One virtue I demonstrated in our senior design work was diligence. This virtue was important to me because it showed a strong commitment to the project's success despite unexpected hardships. I demonstrated this virtue by consistently contributing significant time towards our project throughout the semester. I created an educational resource for the team and developed a stable foundation for our hybrid rendering solution. Additionally, I made myself available to support team members, which fostered collaboration and progress. This helped the project by ensuring the ease of future development for the graphics portion and by providing resources and assistance to team members.

A virtue that was important to me, but that I did not demonstrate as fully in my senior design work, was attentiveness. This virtue was important to me because it ensured that all project aspects were fully addressed and all members were aligned with their goals. I did not demonstrate this virtue fully because I did not consistently hold our team accountable for the soft deadlines we set for ourselves and occasionally prioritized other classes over ensuring the project was progressing as it should. There also needed to be more communication between the graphics and machine learning teams, leading to inefficiencies and misalignments, which could have been addressed proactively. To demonstrate this virtue more effectively in the future, I planned to take a more active role in team management by regularly checking in with all team members and setting more explicit weekly goals to ensure alignment across the team.

7.3.3 Brian Xicon's Virtues

One virtue I demonstrated in our senior design work was responsibility. This virtue was important because it showed that I could be trusted to complete tasks critical to our team's success. I demonstrated this virtue by meeting my deadlines and ensuring the machine learning modules were thoroughly created. This helped the project by maintaining steady progress in our machine learning tasks.

A virtue that was important to me, but I had not demonstrated fully in my senior design work, was foresight. This virtue was significant to me because thinking ahead and anticipating potential problems allowed our team to allocate the necessary time to address them. I recognized that I had not demonstrated this virtue completely because I sometimes focused on immediate tasks and had difficulty giving myself adequate time for certain tasks. To demonstrate this virtue more effectively in the future, I planned to dedicate more time during our sprints to identify possible upcoming challenges, allowing myself adequate time to solve them.

7.3.4 Luke Broglio's Virtues

One virtue I have demonstrated in our senior design work is persistence. This virtue is important to me because I think it is essential to the success of any project and is a strength I value. I have demonstrated this virtue by continuing to actively work on our project even when I encountered complicated or hard-to-diagnose issues. This has helped the project by allowing me to finish demos or project components quicker.

Another virtue I worked to demonstrate, especially in the second semester, was collaboration. At the end of the first semester, I set a goal to collaborate more in order to speed up development. To meet this virtue, I made two steps of note, the first was to have a pair programming session with Kyle so we could better understand each other's code. The other step I took was to increase my participation in code reviews during merge requests.

7.3.5 Ethan Gasner's Virtues

One virtue I have demonstrated in our senior design work thus far is accountability. This virtue is important to me because I believe that if people are held accountable for their actions, it results in a more productive and efficient work environment. I have demonstrated this virtue by ensuring that all documentation assignments and weekly reports are finished and turned in on time. I specifically check on the progress of assignments and notify individuals as a reminder to finish their assigned portion of these documents. I am also the individual responsible for turning in these documents and updating the team website as well. This has helped the project stay on track and ensure documentation is finished on time.

A virtue that was important to me throughout this project, but that I recognized as underdeveloped at the midpoint, was software sustainability. This virtue matters to me because I aim to create software that is not only impactful in the short term but also maintainable and usable over a long period. Earlier in the project, much of our work focused on prototypes and core functionality, with limited attention to long-term maintainability.

Now, at the end of the project, I've made meaningful progress toward demonstrating this virtue. I have taken steps to improve the structure and readability of the codebase by incorporating consistent documentation and meaningful comments. I also worked to modularize key components so that future developers can more easily understand, update, or extend our work. While there is still room for improvement, I believe I have made a strong start toward aligning my work more closely with the value of sustainable software development.

7.3.6 Kyle Kohl's Virtues

One virtue I have demonstrated in our entirety of our senior design work is enthusiasm. This virtue is important to me because I believe that without it, the work becomes a grind, and the motivation to keep working disappears. I have done my best to demonstrate this virtue by encouraging and helping out my team however I can. I can't say I have done this perfectly, but I believe I have done my best.

A virtue that is important to me is responsibility. This virtue is important to me because responsibility is the basis for doing a good job. I am happy to look back and say that I worked hard in this class and gave it the due effort. I was able to attend and participate in all our meetings or give warning otherwise.

8 Conclusions

8.1 Summary of Progress

Our project has made significant strides towards creating a system for novel view synthesis using Gaussian splatting and traditional polygon model rendering. The system is designed to process images or videos from the real world, create a 3D representation through Structure-from-Motion (SfM), convert it into a scene of 3D Gaussians, and render the scene alongside user-defined polygon meshes. We divided the system into two primary components: AI models that generate and enhance the 3D Gaussian scene, and a Unity-based ray-traced renderer that handles real-time rendering.

We've successfully developed a program that integrates 3D Gaussians by reading and loading them to the GPU. We overrode the built-in Unity rendering pipeline by creating a command buffer and inserting the execution at the end of the After Everything Camera Event. The hybrid ray tracer efficiently renders PBR polygon meshes and 3D Gaussian models using multi-intersection and a BVH approach. We have packaged our code in a succinct, easy-to-use package for simple distribution in Unity.

8.2 Value Provided

The core value of this system lies in its seamless integration with Unity, providing an intuitive platform for real-time 3D visualization. By combining Gaussian splatting with traditional polygon mesh rendering, our system offers a hybrid approach that enhances scene detail and realism. The flexibility to render both Gaussians and polygons together within Unity allows for more dynamic and diverse scene creation, making it a powerful tool for users working with complex 3D environments.

The integration with Unity not only facilitates ease of use but also enables users to interact with and modify their scenes in real-time. This combination of Gaussian-based rendering and traditional polygon meshes creates a unique rendering workflow that gives users greater control over the appearance and performance of their 3D scenes. The ability to modify lighting dynamically in Unity further adds to the system's versatility, offering a high degree of customization for visual output.

In terms of performance, the system achieves the target of real-time rendering, maintaining at least 30 frames per second during model rendering, and offers scalable performance across a range of hardware configurations. This makes the system accessible to a broad range of users, from those with high-end hardware to those using more modest systems, with the only restriction being the need for CUDA-compatible hardware when choosing to use the optimizer.

8.3 Next Steps

Although the project has progressed well, we acknowledge that there are still challenges to address, particularly in the areas of model quality and packaging trained machine learning models for Unity integration. Moving forward, the next steps include refining the AI model to improve Gaussian density during the optimizer, which was not accomplished, and further optimizing the raytracer for even faster performance, as well as the inclusion of physically based rendering techniques. Additionally, the task of finalizing the packaging process to ensure that the system is easily distributable as a Unity package still remains. Lastly, further testing and iteration will be required to fully optimize the solution for a broader range of hardware.

While we've made substantial progress towards our goals, there are still opportunities to enhance the system's performance and quality as well as user experience. We're confident that with continued effort, the remaining challenges can be overcome.

9 References

B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," ACM Transactions on Graphics, vol. 42, no. 4, Jul. 4AD, [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Gao, J., et al. "Relightable 3D Gaussian: Real-time Point Cloud Relighting with BRDF Decomposition and Ray Tracing," in arXiv:2311.16043, 2023.

Nicolas Moenne-Loccoz, undefined., et al. "3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes," in ACM Transactions on Graphics and SIGGRAPH Asia, 2024.

10 Appendices

Any additional information that would be helpful to the evaluation of your design document?

If you have any large graphs, tables, or similar data that do not directly pertain to the problem but help support it, include it here. This would also be a good area to include the hardware/software manuals used. May include CAD files, circuit schematics, layout, etc, PCB testing issues, etc., Software bugs, etc.

Appendix 1 – Operation Manual

Prerequisites before use:

Software Requirements:

- Must have Python 3.10+
- Must have the plyfile library installed
- Must have the ipython library installed
- Must have the torch library installed
- Must have opencv-python library installed
- Must have torchvision library installed
- Must have tkinter library installed. (*tkinter must be installed globally*)
- Must have Unity installed on your computer.
- Must have CUDA installed on your computer.
- Must have Git installed on your computer.

Hardware Requirements:

- Must have an NVIDIA graphics card capable of running CUDA for optimizing 3D Gaussians.

Installation

1. Install Python

Follow these instructions for Windows.

Follow these instructions for Mac.

1. Download Python Executable Installer
2. Run Executable Installer
3. Add Python to PATH during installation
4. Verify Python is installed on Windows
5. Verify pip is installed on Windows

2. Install tkinter

For Windows

Run this command in your terminal

```
pip install tkinter
```

For Linux

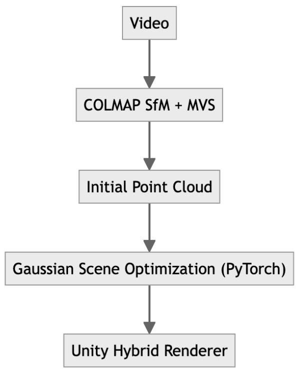
```
sudo apt-get install python3-tk
```

3. Install other dependencies

Run this command in your terminal:

pip install plyfile ipython torch torchvision opencv-python

Pipeline Overview



Features

- Seamless Structure-from-Motion integration
- Convert video into a fully modeled 3D scene
- Hybrid Triangle-Gaussian-based scene rendering
- Unity-compatible real-time ray tracer

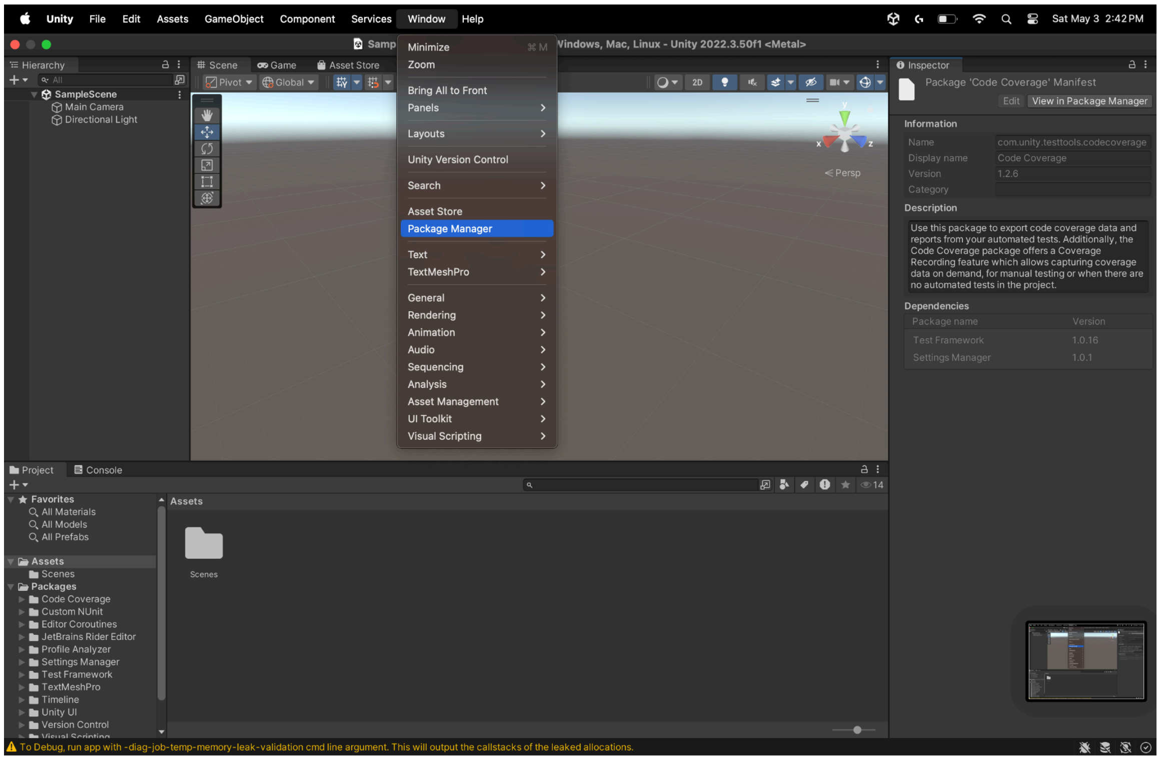
File Structure

```
Assets/
└── Hybrid Relightable 3D Gaussian Rendering/
    ├── Scripts/
    ├── Prefabs/
    ├── Materials/
    ├── Textures/
    ├── Scenes/
    └── Documentation/
```

Usage

Setup Project

1. Start a new Unity project
2. Open the **Package Mangager**
   - It can be found within the **Window** tab.



Information

Name	com.unity.testtools.codecoverage
Display name	Code Coverage
Version	1.2.6
Category	

Description

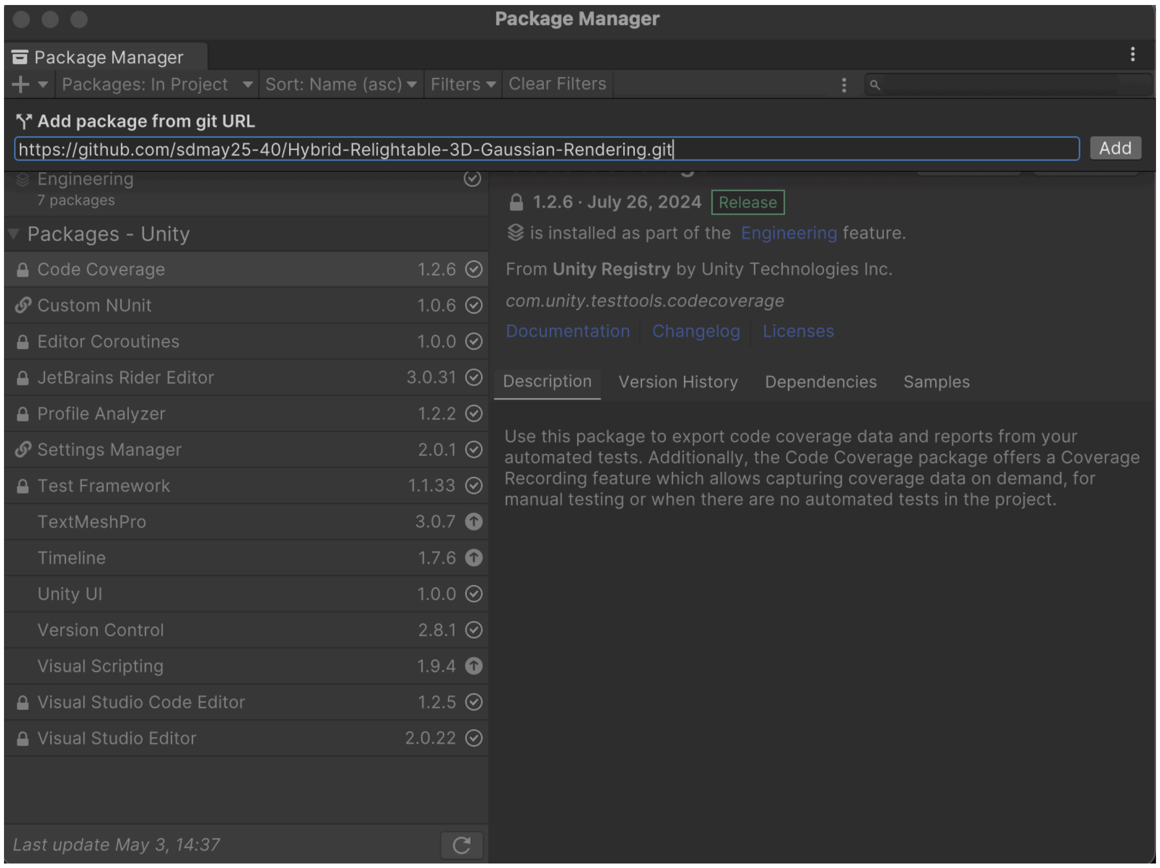
Use this package to export code coverage data and reports from your automated tests. Additionally, the Code Coverage package offers a Coverage Recording feature which allows capturing coverage data on demand, for manual testing or when there are no automated tests in the project.

Dependencies

Package name	Version
Test Framework	1.0.16
Settings Manager	1.0.1

To Debug, run app with -diag-job-temp-memory-leak-validation cmd line argument. This will output the callstacks of the leaked allocations.

3. Add the package from git
   1. Click on the '+' symbol towards the top of the window
   2. Click Add Package from git URL
   3. Enter the URL https://github.com/sdmay25-40/Hybrid-Relightable-3D-Gaussian-Rendering.git.



https://github.com/sdmay25-40/Hybrid-Relightable-3D-Gaussian-Rendering.git

Code Coverage	1.2.6
Custom NUnit	1.0.6
Editor Coroutines	1.0.0
JetBrains Rider Editor	3.0.31
Profile Analyzer	1.2.2
Settings Manager	2.0.1
Test Framework	1.1.33
TextMeshPro	3.0.7
Timeline	1.7.6
Unity UI	1.0.0
Version Control	2.8.1
Visual Scripting	1.9.4
Visual Studio Code Editor	1.2.5
Visual Studio Editor	2.0.22

1.2.6 · July 26, 2024

is installed as part of the Engineering feature.

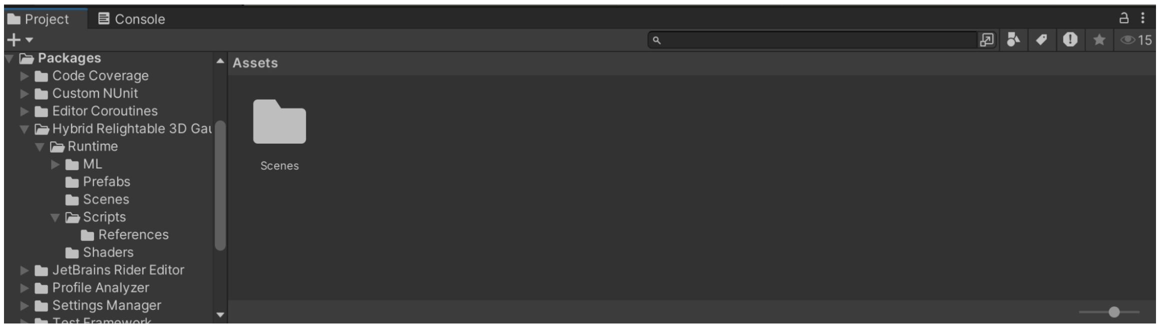
From **Unity Registry** by Unity Technologies Inc.

com.unity.testtools.codecoverage

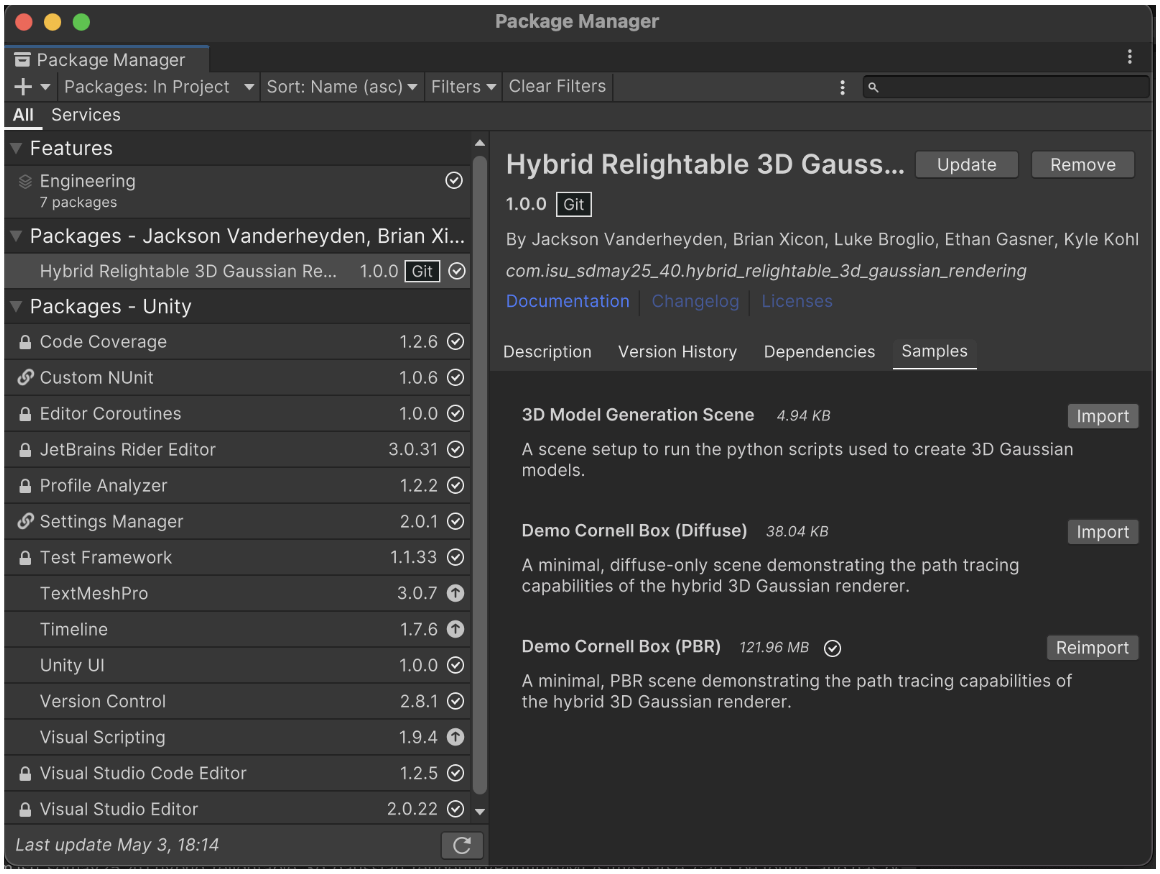
Use this package to export code coverage data and reports from your automated tests. Additionally, the Code Coverage package offers a Coverage Recording feature which allows capturing coverage data on demand, for manual testing or when there are no automated tests in the project.

Last update May 3, 14:37

2. The project should now be able to be found in the Packages folder of the project explorer.



3. Import the 3D Model Generation Scene from the Samples tab of the package.


Package Manager

Packages: In Project | Sort: Name (asc) | Filters | Clear Filters

All | Services

Features

Engineering 7 packages		✓

Packages - Jackson Vanderheyden, Brian Xi...

Hybrid Relightable 3D Gaussian Re...	1.0.0 Git	✓

Packages - Unity

Code Coverage	1.2.6	✓
Custom NUnit	1.0.6	✓
Editor Coroutines	1.0.0	✓
JetBrains Rider Editor	3.0.31	✓
Profile Analyzer	1.2.2	✓
Settings Manager	2.0.1	✓
Test Framework	1.1.33	✓
TextMeshPro	3.0.7	↑
Timeline	1.7.6	↑
Unity UI	1.0.0	✓
Version Control	2.8.1	✓
Visual Scripting	1.9.4	↑
Visual Studio Code Editor	1.2.5	✓
Visual Studio Editor	2.0.22	✓

Last update May 3, 18:14

Hybrid Relightable 3D Gauss... Update | Remove

1.0.0 Git

By Jackson Vanderheyden, Brian Xicon, Luke Broglio, Ethan Gasner, Kyle Kohl

com.isu_sdmay25_40.hybrid_relightable_3d_gaussian_rendering

Documentation | Changelog | Licenses

Description | Version History | Dependencies | Samples

3D Model Generation Scene *4.94 KB* Import

A scene setup to run the python scripts used to create 3D Gaussian models.

Demo Cornell Box (Diffuse) *38.04 KB* Import

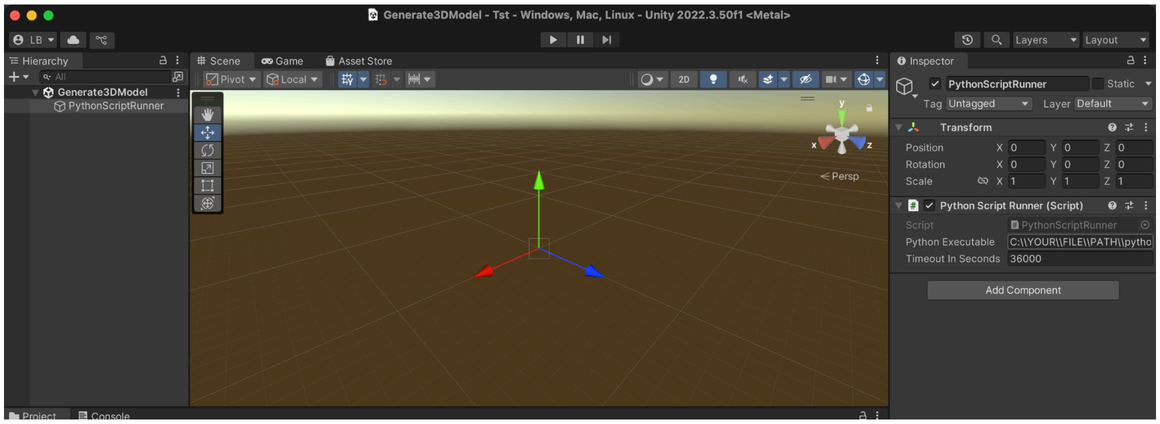
A minimal, diffuse-only scene demonstrating the path tracing capabilities of the hybrid 3D Gaussian renderer.

Demo Cornell Box (PBR) *121.96 MB* ✓ Reimport

A minimal, PBR scene demonstrating the path tracing capabilities of the hybrid 3D Gaussian renderer.

Build Gaussian Model with machine learning

1. Open the 3D Model Generation Scene from the Project Explorer
2. In the python script runner object set the Python script runner attribute to be the path to your python executable. ‘


Generate3DModel - Tst - Windows, Mac, Linux - Unity 2022.3.50f1 <Metal>

Hierarchy: Generate3DModel > PythonScriptRunner

Inspector

PythonScriptRunner | Static

Tag Untagged | Layer Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Python Script Runner (Script)

Script	PythonScriptRunner
Python Executable	C:\\YOUR\\FILE\\PATH\\pytho
Timeout In Seconds	36000

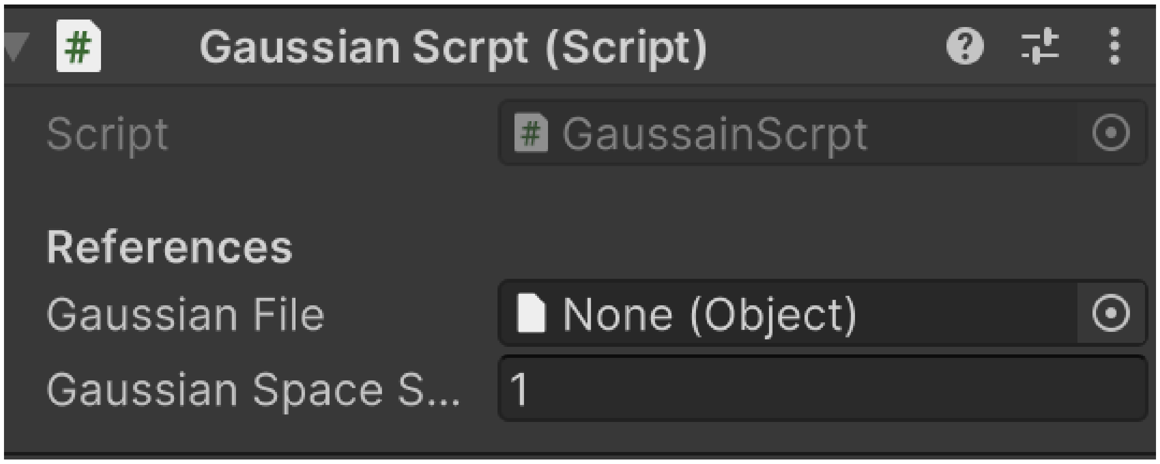
Add Component

3. Run the unity scene
4. When prompted, select the video files you desire to turn into a 3D model.

5. The progress of the render is outputted to the Unity Console.

Renderer Setup

1. Add the Main Camera prefab from the Runtime/Prefabs folder of the Hybrid Relightable 3D Gaussian package to your scene.
    - Remove any other cameras.
2. For every Gaussian model you would like to render add a GaussianModel prefab from the Runtime/Prefabs folder.
    - Then drag the .ply model to the Gaussian File attribute.
    - You might need to adjust the Gaussian Space Scale attribute based on the gaussians look in the scene.



Gaussian Scrpt (Script)

Script: GaussainScrpt

References

Gaussian File: None (Object)

Gaussian Space S...: 1

3. Add any polygon models you want to render to the scene.
4. Run the scene by pressing the play button.

Appendix 2 – alternative/initial version of design

- Vulcan or GLSL Version
    - We initially considered building our own engine using a graphics API such as vulcan or GLSL but settled upon Unity so we could focus on the renderer itself more.
- Full Relightable Gaussian Version
    - Our original goal was for our machine learning model to extra the normal vectors and PBR properties of Gaussians so they could have their lighting data changed in real time. This had to be removed due to time constraints.

Appendix 3 – Other considerations

- Overall, the team rose up to meet the challenges of this project. We all grew a lot in our skills repertoire to make this project come to fruition. The firsthand experience of seeing a project come from start to finish has impacted the whole team and we will proudly bring that experience with us to our next team.

- Our project can be added to unity with this Git link: https://github.com/sdmay25-40/Hybrid-Relightable-3D-Gaussian-Rendering.git.

Appendix 4 – Code

- Code on Git Lab

Appendix 5 – Team Contract

Complete each section as completely and concisely as possible. We strongly recommend using tables or bulleted lists when applicable.

Team Members

Team Members:

1) _____Jackson Vanderheyden_______ 2) ______Brian Xicon_____________

3) _____Ethan Gasner ______________ 4) ______Kyle Kohl ____________

5) _____Luke Broglio _______________

Required Skill Sets for Your Project

(If feasible, tie them to the requirements.)

Skill Sets covered by the Team

1. The skills, expertise, and unique perspectives Jackson Vanderheyden brings to the team are: computer graphics industry experience, multiple Unity projects, & previous experience implementing ray tracing.
2. The skills, expertise, and unique perspectives Brian Xicon brings to the team are: C, C++,HTML, CSS, JS, Python, PyTorch experience, Machine Learning experience, and Computer Vision experience.
3. The skills, expertise, and unique perspectives Ethan Gasner brings to the team are: C, C++, python, AI Experience, Machine learning experience, Unity Experience, unique Cyber security perspective, JavaScript, HTML/CSS. Additionally, a cooperative and easygoing attitude.
4. The skills, expertise, and unique perspectives Kyle Kohl brings to the team are: C, C++, Java, Python, a little bit of HTML and CSS. He has the definite advantage of being an extrovert that loves to encourage others. Lots of experience in the communication role.
5. The skills, expertise, and unique perspectives Luke Broglio brings to the team are: Experience with C, C++, python, graphics programming, Unity, HTML/CSS, Javascript, writing a raytracer and experience with working in agile/scrum development environments.

Project Management Style Adopted by the Team

Typically, Waterfall or Agile for project management.

Individual Project Management Roles

(Enumerate which team member plays what role.)

1 Team Contract

Team Members:

1) _____Jackson Vanderheyden_______ 2) ______Brian Xicon_____________

3) _____Ethan Gasner ______________ 4) ______Kyle Kohl ____________

5) _____ Luke Broglio _______________

Team Procedures

1) Face to face Friday at 9:30AM serves as our regular team meeting schedule. Additionally, Meet on Tuesdays at 4pm biweekly with our advisor: Professor Mitra

2) Discord will be used as our preferred method of communication for updates, reminders, issues, and scheduling.

3) Consensus based decision making policy will be employed, ensuring all members contribute to the discussion.

4) The Communication Manager will take primary responsibility for meeting record keeping. All records will be stored on the relevant Figma page.

5) Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

6) Decision-making policy (e.g., consensus, majority vote):

7) Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Participation Expectations

1. All team members must punctually attend (i.e., be no more than 10 minutes tardy) and participate in all meetings. All absences must be excused 24 hours prior to the set meeting time.
2. All team members are expected to complete their assigned work prior to the agreed upon deadline.
3. All team members must continuously and accurately report progress on the Kanban board and in meetings. Unexpected delays resulting in timeline changes must be reported as soon as possible.

4. All team members are expected to provide feedback on all team decisions and commit the necessary number of hours outside of class time to complete assigned work before the agreed upon deadline.

Leadership

1. Discord will be used as an informal Q&A forum to help support and guide the work of all team members.

2. Team members should provide positive peer to peer feedback to recognize the contributions of all team members.

3. The following roles serve as general, not rigid, guidelines, and therefore, work traditionally specified for a particular managerial role was not exclusively designated to that individual.

4. Kyle will serve as the Communication Manager. Their duties include serving as the primary point of contact for all stakeholder communication, scheduling meetings, and recording relevant meeting information (e.g., attendance, meeting content, etc.).

5. Ethan Gasner will serve as the Documentation Manager. Their duties include maintaining all living documentation (e.g., project charter, product models, etc.) related to the project and leading primary direction on the project website.

6. Jackson Vanderheyden will serve as the Graphics Scope Manager. Their duties include identifying field specific functional and nonfunctional requirements and assisting the Schedule Manager in formulating an accurate timeline.

7. Brian Xicon will serve as the Machine Learning Scope Manager. Their duties include identifying field specific functional and nonfunctional requirements and assisting the Schedule Manager in formulating an accurate timeline.

8. Luke Broglio will serve as the Schedule Manager. Their duties include communicating with the Graphics & Machine Learning Scope Managers to create timelines and deliverables and orchestrate development resources through sprints.

Collaboration and Inclusion

6. The skills, expertise, and unique perspectives Jackson Vanderheyden brings to the team are: computer graphics industry experience, multiple Unity projects, & previous experience implementing ray tracing.
7. The skills, expertise, and unique perspectives Brian Xicon brings to the team are: C, C++,HTML, CSS, JS, Python, PyTorch experience, Machine Learning experience, and Computer Vision experience.
8. The skills, expertise, and unique perspectives Ethan Gasner brings to the team are: C, C++, python, AI Experience, Machine learning experience, Unity Experience, unique Cyber security perspective, JavaScript, HTML/CSS. Additionally, a cooperative and easygoing attitude.
9. The skills, expertise, and unique perspectives Kyle Kohl brings to the team are: C, C++, Java, Python, a little bit of HTML and CSS. He has the definite advantage of being an extrovert that loves to encourage others. Lots of experience in the communication role.

10. The skills, expertise, and unique perspectives Luke Broglio brings to the team are: Experience with C, C++, python, graphics programming, Unity, HTML/CSS, Javascript, writing a raytracer and experience with working in agile/scrum development environments.
11. Strategies for encouraging and supporting contributions and ideas from all team members: Through the following, Create a safe and open Environment, Use of Structured Discussion, Leverage Team Diversity. Ask Open Ended Questions, Active listening and Acknowledgment.
12. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?) Our group will be very open with each other, if any of us have any concerns with the environment all of us will be open to feedback and work together to discover steps to resolve the issue.

Goal-Setting, Planning, and Execution

1. Team goals for this semester: Have a working prototype raytracer that uses 3D Gaussians

2. Strategies for planning and assigning individual and team work: assign tasks primarily based on individual role then assign tasks based on interest and best fit.

3. Strategies for keeping on task: General time management techniques based on individuals.

Consequences for Not Adhering to the Team Contract

1. If an individual does not adhere to the above standards, We will start off with verbal warnings from the team and potentially a meeting if necessary.

2. If this continues to become an issue we will bring this up with the professors and course coordinator.

****

a. I participated in formulating the standards, roles, and procedures as stated in this contract.

b. I understand that I am obligated to abide by these terms and conditions.

c. I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) _______Jackson Vanderheyden_____________________ DATE _______9/13/2024______

2) _______Brian Xicon______________________________ DATE _______9/13/2024______

3) _______Ethan Gasner_____________________________ DATE _______9/13/2024______

4) _______Kyle Kohl_________________________________ DATE ______9/13/2024______

5) _______Luke Broglio_____________________________ DATE _____9/13/2024______