Kindt Bobbie

Hybrid spectral-RGB rendering

A PIPELINE FOR REAL-TIME GRAPHICS APPLICATIONS

Supervisor: Samyn Koen

Coach: Cnockaert Helena

Graduation Work 2025-2026

Digital Arts and Entertainment

Howest.be

Kindt Bobbie

CONTENTS

ABSTRACT & KEY WORDS

(ENG)

Physically based rendering (PBR) aims to simulate the transport of light in a way that is both visually convincing and physically accurate. Real-time and offline renderers commonly rely on RGB color representations, which approximate light using three fixed channels. While it’s efficient in terms of performance and memory, this approximation limits the accuracy of wavelength-dependent simulations such as dispersion, spectral absorption, and other complex material-light interactions. Spectral rendering addresses these limitations by representing light as a continuous function over wavelengths, but it does introduce significant computational and implementation challenges.

This paper will analyze and compare a real-time renderer with an RGB, spectral and hybrid pipeline. This hybrid pipeline is a combination of both spectral rendered and RGB rendered assets in a scene. With this I want to render specific assets physically accurately where it matters while taking advantage of the advantages RGB pipelines have in terms of performance.

(NED)

Physically based rendering (PBR) heeft als doel het transport van licht te simuleren op een manier die zowel visueel overtuigend als fysisch nauwkeurig is. Real-time en offline renderers maken doorgaans gebruik van RGB-kleurweergaven, die licht benaderen met behulp van drie vaste kanalen. Hoewel het efficiënt is qua prestaties en geheugen, beperkt deze benadering de nauwkeurigheid van golflengteafhankelijke simulaties zoals dispersie, spectrale absorptie en andere interacties tussen materiaal en licht. Spectrale rendering lost deze beperking op door licht weer te geven als een continue functie over golflengten, maar dit brengt wel aanzienlijke uitdagingen met zich mee op vlak van berekeningen en implementatie.

In deze paper wordt een real-time renderer met een RGB-, spectrale en hybride gebaseerde weergave besproken en vergeleken. Deze hybride gebaseerde weergave is een combinatie van zowel spectraal gerenderde als RGB-gerenderde objecten in een scene. Hiermee wil ik specifieke objecten fysiek nauwkeurig kunnen renderen waar dat belangrijk is, terwijl ik profiteer van de voordelen die RGB-pijplijnen bieden op vlak van prestaties.

PREFACE

This work was started as part of a broader interest in physically based rendering and graphics engine development. Through both academic coursework, and personal projects and interests, it became apparent that many visually striking and physically meaningful optical effects like dispersion and polarization are either approximated, faked, or entirely excluded from conventional renderers due to their reliance on three RGB channel color representation.

The motivation for this project stems from an interest in exploring spectral rendering not only as a theoretical concept but as a practical system that could be integrated into a modern rendering pipeline. While spectral rendering is often regarded as computationally expensive or impractical for interactive applications, recent research and open-source implementations suggest that this assumption is no longer fully true. At the same time, hybrid rendering approaches have demonstrated that combining multiple rendering techniques can yield favorable trade-offs between performance and visual accuracy.

This project aims to bridge these two domains by exploring how spectral light transport can be incorporated into a hybrid renderer.

LIST OF FIGURES

INTRODUCTION

Physically Based Rendering (PBR) aims to simulate the interaction of light with surfaces in a scene according to the laws of physics. The most important aspect of this approach is the accurate simulation of light transport, which is formalized through the rendering equation. By accounting for emitted, reflected, and incoming radiance at each point in a scene, the rendering equation provides a theoretical framework for producing realistic light behavior.

In practice, however, many rendering systems approximate light transport by representing color using three channels: Red, Green, and Blue (RGB). This representation offers significant computational advantages and aligns well with display hardware, but it fundamentally reduces to a low approximation of the continuous spectral nature of light. As a result, RGB-based pipelines struggle to accurately reproduce wavelength-dependent phenomena such as chromatic dispersion, wavelength-specific absorption, and subtle variations in material appearance under different illuminants.

Against RGB rendering, spectral rendering addresses these limitations by modeling light as a function of wavelength rather than discrete (RGB) color channels. This enables more physical and accurate results in comparison to RGB rendered scenes. Despite its advantages, spectral rendering has generally been considered too costly for widespread adoption, particularly in real-time or interactive contexts due to its increased computational cost, higher memory requirements and complex sampling strategies. Therefore, most spectral rendering research focuses on offline rendering whereas real-time rendering systems continue to rely on RGB-based approaches.

Recent research challenges the long-standing assumption that spectral rendering is prohibitively expensive for interactive or real-time applications by proposing efficient spectral sampling strategies, compact spectral representation, and GPU-friendly implementations. In parallel, hybrid rendering approaches have gained prominence by combining multiple rendering techniques, such as rasterization and raytracing, to balance performance and visual accuracy. These developments suggest new opportunities for integrating spectral rendering into practical rendering pipelines by treating it as a selectively applied enhancement rather than a globally enforced rendering mode.

The motivation for this research stems from the observation that recent advantages in GPU hardware, real-time ray tracing support, and hybrid rendering techniques have significantly narrowed the performance gap between physically accurate rendering techniques and real-time limitations. Simultaneously, modern game engines increasingly rely on hybrid rendering pipelines that combine rasterization, ray tracing, and post-processing techniques to target expensive computations to specific materials, lighting interactions, or screen regions where they provide the greatest visual benefit. This raised the question of whether spectral rendering too could be integrated into such pipelines in a similar selective manner, enabling improved color accuracy and wavelength-dependent effects without compromising real-time performance.

This literature study examines foundational concepts in radiometry and light transport, surveys key research in spectral rendering, and reviews practical implementations that bridge theory and application. Additionally, it investigates spectral rendering within the broader landscape of hybrid rendering systems. These works form the theoretical and technical foundation for the hybrid spectral renderer presented in this paper.

My research question with its corresponding hypothesis for this paper are:

“**How can a hybrid spectral-RGB rendering pipeline be designed to optimize the trade-off between perceptual color accuracy and real-time rendering performance in modern game engines?”**

H0: Implementing selective spectral sampling in a hybrid spectral-RGB pipeline results in a mean color deviation (ΔE) below 5% compared to full spectral rendering, while sustaining a frame rate of at least 60 FPS on modern GPUs.

H1: Limiting spectral computations to reflective and refractive materials yields a higher ratio of visual accuracy to performance cost compared to applying spectral rendering uniformly across all materials.

LITERATURE STUDY / THEORETICAL FRAMEWORK

To establish the theoretical foundation for spectral rendering, this chapter will briefly address the physical and mathematical principles for light transport in rendering systems. Section 1.1 reviews the rendering equation as the core formulation of physically based rendering. Section 1.2 then distinguishes between radiometric and photometric descriptions of light, highlighting the importance of wavelength-dependent formulations for physical accuracy. Section 1.3 places these concepts within the context of the visible portion of electromagnetic spectrum, clarifying which wavelengths are relevant to image synthesis and human perception. Finally, section 1.4 discusses the limitations of RGB-based light transport models, motivating the need for spectral approaches.

1. INTRODUCTION TO RENDERING AND LIGHT TRANSPORT

Rendering aims to generate images by simulating the transport of light through a virtual scene. The physical behavior of light is commonly described using radiometric quantities such as radiance, irradiance, and spectral power distributions[1]. This together defines how energy is emitted, reflected, and absorbed by surfaces. These principles are unified by the rendering equation (See Fig. 1). As described by [11] Delombaerde and Vandenberghe, the rendering equation encapsulates the interaction of light with surfaces and forms the theoretical foundation in modern rendering systems.

$$L_o(X,\vec{\omega}_{o,x}) = \underset{\text{Object}}{L_e(X,\vec{\omega}_{o,x})} + \int \underset{\text{Lights}}{L_i(X,\vec{\omega}_{i,x})}\, \underset{\text{Materials}}{f_{X,n}(\vec{\omega}_{i,x},\vec{\omega}_{o,x})}\, \underset{\text{View Angle}}{\cos\theta_i}\, \mathrm{d}\omega_{i,x}$$

Figure 1: The rendering equation [11]

1.1. THE RENDERING EQUATION

The rendering equation expresses the outgoing radiance at a surface point as the sum of emitted radiance and reflected incoming radiance over the hemisphere of incident directions. By recursively accounting for light interactions between surfaces, the equation models both direct and indirect illumination and this captures global lighting effects. The rendering equation serves as the basis for numerical solution techniques such as Monte Carlo integration (see further in 2.2.5), which are used in lots of modern rendering algorithms. [30]

[1] Spectral power distribution is a measurement that describes the proportion of total light that is emitted, transmitted or reflected by a color sample at each visible wavelength. It often refers to the concentration of any radiometric or photometric quantity like radiance, irradiance, radiant energy, … [16][23]

1.1.1. BIDIRECTIONAL REFLECTANCE DISTRIBUTION FUNCTION (BRDF)

The interaction between incoming and outgoing light at a surface is commonly described using a Bidirectional Reflectance Distribution Function (BRDF). The BRDF defines how much radiance arriving from one direction is reflected towards another direction, and it encodes material properties such as roughness, reflectivity, and anisotropy.

In the rendering equation, the BRDF is in control of how incident radiance contributes to outgoing radiance at a surface point. While traditional RGB-based renderers evaluate BRDFs using color triples, physically based and spectral rendering approaches extend BRDF evaluations to operate on wavelength-dependent reflectance functions.

1.2. PHOTOMETRY VS RADIOMETRY

Radiometry provides a wavelength-dependent description of light energy, independent of human perception. Photometry incorporates the response of the human visual system by weighting radiometric quantities according to perceptual sensitivity. Peters [21, 22] emphasizes that although photometric quantities are useful for perception-oriented evaluation and display, physically accurate rendering fundamentally relies on radiometric formulations, which are inherently spectral in nature.

Light sources and materials interact differently across wavelengths, influencing occurrences such as absorption, scattering, and reflection. If we collapse this behavior into a small set of color channels, it introduces approximation errors. As a result, radiometric formulations are essential when the goal is to simulate light transport in a physically accurate technique rather than merely reproducing perceptual appearances.

1.3. VISIBLE LIGHT AND THE ELECTROMAGNETIC SPECTRUM

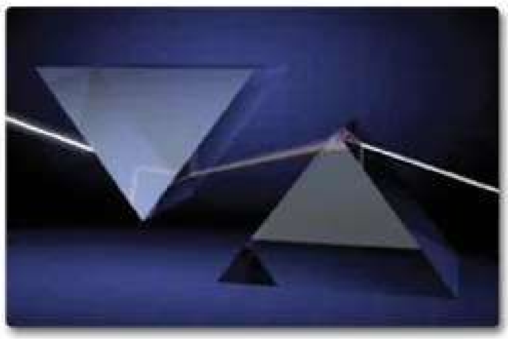
Light is a form of electromagnetic radiation characterized by its wavelength or frequency. The electromagnetic spectrum spans a wide range of wavelengths, from gamma rays to radio waves. Human vision, however, is sensitive to only a narrow portion of this spectrum known as visible light. According to NASA [33], visible light occupies wavelengths approximately between 380 and 750 nanometers (nm), ranging from violet at shorter wavelengths (400 nm) to red at longer wavelengths (700 nm).

Within this visible range, different wavelengths correspond to different perceived colors. However, light sources and materials typically emit and reflect energy across continuous spectral distributions rather than discrete bands. As a result, two light sources with distinct spectral power distributions can produce the same perceived color, a phenomenon known as metamerism [2]. This property is central to understanding both the strengths and limitations of RGB-based rendering models.

Because light interactions are wavelength-dependent, physically based simulations must account for spectral variations. Although photometric quantities incorporate the human eye's sensitivity, the physical interactions

[2] Metamerism is the perceived matching of colors with different spectral power distributions. [16]

between light and matter remain inherently wavelength-dependent. This motivates spectral rendering approaches, which model light transport across the visible spectrum rather than approximating it with a small set of color channels.



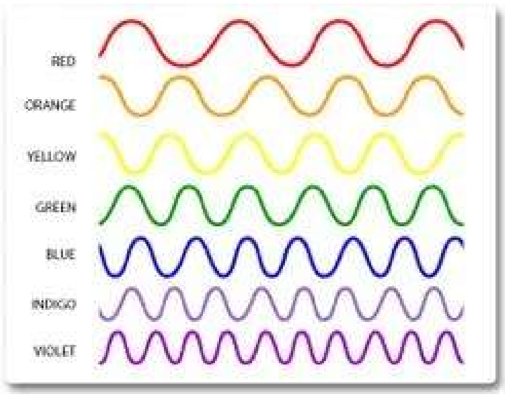


Figure 2: (Left) Isaac Newton's experiment in 1665 showed that a prism bends visible light andthat each color refracts at a slightly different angle depending on the wavelength of the color. Credit: Troy Benesch. (Right) Each color in a rainbow corresponds to a different wavelength of electromagnetic spectrum. [33]

1.4. LIMITATIONS OF RGB-BASED LIGHT TRANSPORT

Traditional real-time rendering pipelines often bypass these complexities by representing radiance using RGB color triplets. While RGB-based rendering is efficient in terms of increased performance gain and memory, this approach assumes that three values are sufficient to approximate the spectral distribution of light. Peercy [14] already highlighted the limitations of linear RGB representations for accurately modeling spectral interactions, noting that different spectra can map to identical RGB values.

These observations underline that light transport is fundamentally spectral in nature, and that RGB-based rendering forms a practical approximation rather than a physically complete solution. This motivates the study of spectral rendering techniques, which attempt to more accurately model light transport by working directly in the wavelength domain.

2. SPECTRAL RENDERING

This chapter analyzes spectral rendering as an extension of physically based rendering, focusing on its theoretical motivation, core techniques, benefits, and practical limitations. Section 2.1 introduces spectral rendering and motivates its use by contrasting it with traditional RGB-based approaches. Section 2.2 then reviews classic spectral techniques. Building on this foundation, section 2.3 discusses the key advantages of spectral rendering in terms of physical correctness, perceptual accuracy, and robustness under different lighting conditions. Finally, section 2.4 examines the computational and practical challenges associated with spectral rendering, particularly in real-time contexts.

2.1. DEFINITION & MOTIVATION

Spectral rendering refers to rendering techniques that represent light as a function of wavelength rather than distinct color channels. Instead of storing radiance as RGB values, spectral renderers operate on spectral power distributions, enabling physically accurate simulation of wavelength-dependent effects like dispersion.

Yamaguchi et al. [3] argue that RGB imaging systems are fundamentally limited in their ability to capture and reproduce spectral information. Especially in scenarios involving non-standard illuminants or materials with complex spectral reflectance. Similarly, Croisant [35] demonstrates that RGB renderers fail to accurately reproduce effects such as dispersion, colored shadows and subtle material color shifts under varying lighting conditions.

Peters [26] provides a comparative analysis of RGB and spectral rendering, showing that RGB renderers can be visually convincing in many cases. They start to break down in scenes involving refractive materials, narrowband [a] light sources, or fluorescence. Spectral rendering addresses these issues by maintaining wavelength information throughout the rendering pipeline. This results in improved physical accuracy and perceptual accuracy.

The primary motivation for spectral rendering is therefore not merely visual realism, but visual correctness. By calculating light transport in a way that aligns with reality, spectral rendering avoids many of the approximation errors and artifacts that RGB pipelines produce.

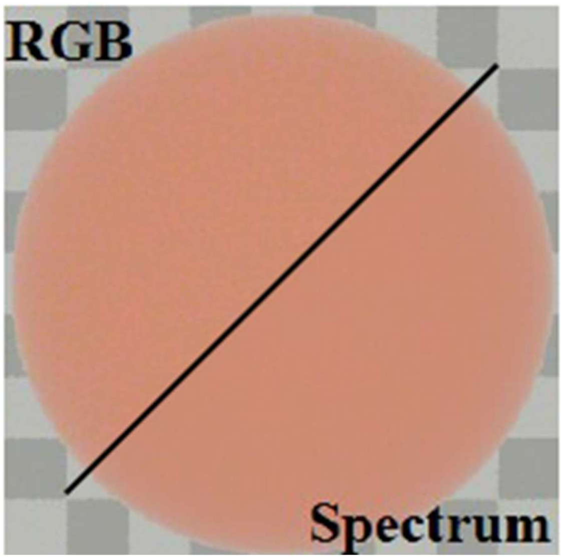
2.2. CLASSIC SPECTRAL RENDERING TECHNIQUES

Spectral rendering techniques aim to model light transport as a function of wavelength rather than discrete color channels. Over time, multiple approaches have been proposed to balance physical accuracy against computational costs. This section reviews the most influential spectral rendering techniques that form the foundation for modern spectral renderers. The techniques I will be discussing are full spectral path tracing, hero wavelength, moment-based and constrained spectral uplifting, the rendering of polarization, dispersion, and fluorescence and lastly how the Monte Carlo technique can be used in the spectral domain.

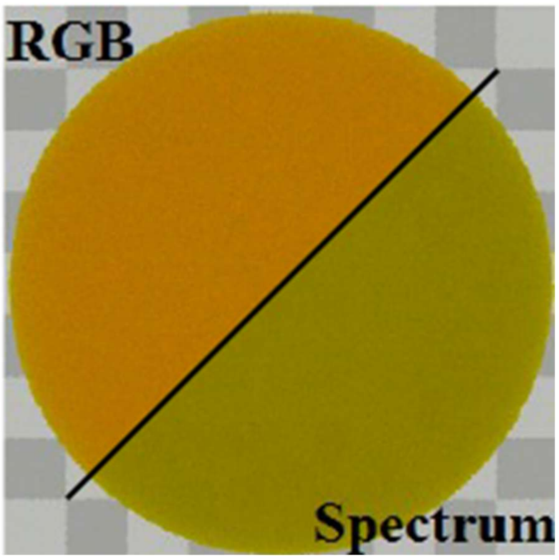
2.2.1. FULL SPECTRAL PATH TRACING

Early approaches to spectral rendering focused on full spectral path tracing. With full spectral path tracing, the radiance is evaluated across a dense set of wavelengths at every interaction between light and a surface. In this formulation, the rendering equation is extended to explicitly operate in the spectral domain. This results in highly accurate simulations of wavelength-dependent interactions.

Radziszewski et al. [2] describe techniques for full spectral rendering that achieve high physical accuracy by sampling radiance across the visible spectrum (see 1.3). However, this technique approach significantly increases computational cost due to the added spectral dimension. This leads to higher memory usage, increased sampling requirements, and longer convergence times. This leads to full spectral path tracing to typically be restricted to offline rendering applications.



Copper sphere illuminated by a D65 white light.



Copper sphere illuminated by a triangular spectral distribution stretched from 535nm to 595nm.

Figure 3: Top left half: an RGB model with 645nm, 526nm and 444nm wavelengths. Right bottom half: our full spectral model. For clarity, only diffuse reflection is calculated. [2]

2.2.2. HERO WAVELENGTH

A more performant technique in comparison to the full spectral path tracing is the hero wavelength technique. Wilkie et al. [12] proposes sampling a single wavelength per light transport path instead, while using importance sampling[3] to approximate the full spectrum.

This approach drastically reduces computational cost while retaining the ability to reproduce wavelength-dependent effects. Mojzík [8] extends the hero wavelength framework to support fluorescence computations. This is achieved by allowing wavelength shifts during light transport and demonstrates that complex spectral effects can still be captured efficiently. Hero wavelength techniques can capture complex spectral effects at a reduced performance cost compared to full spectral methods.

[3] Importance Sampling (MIS) is a Monte Carlo approximation method, and it used to estimate an assumption. It uses the Monte Carlo sampling method to take the average of all samples and them estimate the assumption.

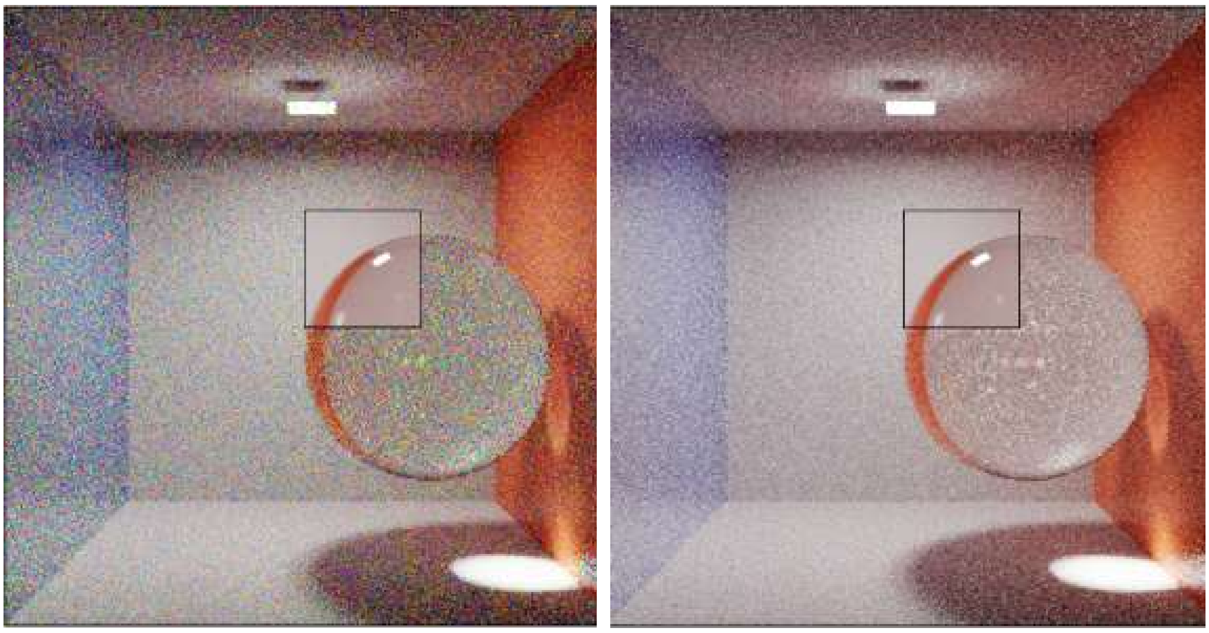


Figure 4: renders of a photon map scene with a non-dispersive glass sphere. 4 samples per pixel, inset: 256 samples. To render the hero wavelength version, only a quarter of the photons used for the single wavelength scene was shot: due to the increased ability of the renderer to find photons that can be used, no decrease in quality can be seen. [12]

2.2.3. MOMENT-BASED AND CONSTRAINED SPECTRAL UPLIFTING

Another class of techniques focuses on compact spectral representations rather than explicit wavelength sampling. Moment-based and constrained spectral uplifting methods reconstruct plausible spectral distributions from RGB or low dimensional input data.

Tódova et al. [17] and Peters and al. [32] show that moment-based representations can efficiently approximate bounded spectral signals while still preserving important spectral data. Mallet and Yuksel [24] further explore spectral primary decomposition which represents spectra using a small set of basis functions optimized for reconstruction quality.

2.2.4. RENDERING OF POLARIZATION, DISPERSION, AND FLUORESCENCE

Beyond general light transport, spectral rendering has been applied to specific optical effects that cannot be accurately represented in RGB pipelines. Wilkie et al. [5] presents a framework for combined rendering of polarization and fluorescence. This is achieved by extending the rendering equation to account for additional physical dimensions.

Similarly, Fontas [7] demonstrates the simulation of chromatic dispersion using spectral sampling. This work shows how visually significant these improvements are over RGB-based approximations. Both works highlight the meaningful power of spectral rendering, but they also emphasize the increased computational demands associated with simulating such effects accurately.

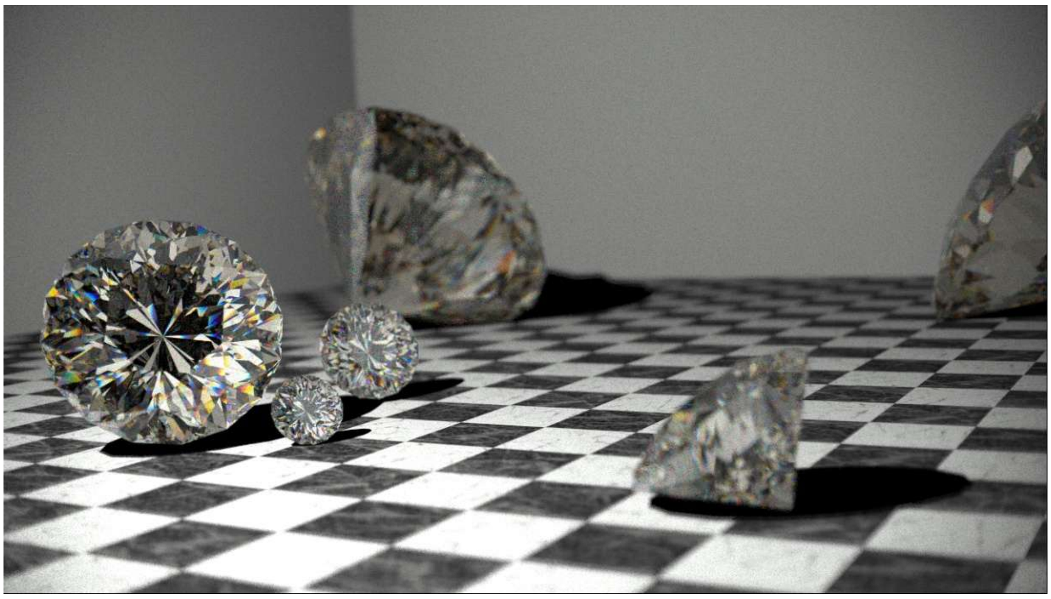


Figure 5: This scene was rendered using a double gaussian lens system with moderate depth of field effects. Each diamond is a transformed instance of the same mesh with each instance sharing the same set of vertices in memory. [7]

2.2.5. MONTE CARLO

Most spectral rendering techniques rely on the Monte Carlo integration to numerically approximate the rendering equation. Monte Carlo methods estimate high dimensional integrals[4] through random sampling and are widely used in physically based rendering due to their flexibility and scalability.

In spectral rendering, wavelengths are treated as an additional sampling dimension alongside spatial position, direction, and time. Peters [21, 22] emphasizes that spectral rendering does not fundamentally alter the structure of a Monte Carlo renderer but rather extends the sampling space to include wavelength-dependent behavior. Radiance is then typically evaluated at one or more sampled wavelengths per path, and the full spectral response is then reconstructed through accumulation and averaging.

While this approach makes spectral rendering manageable, it does introduce additional variance. This particularly shows when simulating narrowband phenomena such as dispersion or fluorescence. The reliance on Monte Carlo integration unifies the spectral rendering techniques mentioned in this section. This insight is very relevant for real-time and hybrid rendering systems, where Monte Carlo ray tracing is already employed for effects such as reflections and global illumination.

[4] High dimensional integrals are an accumulation over a space with lots of variables/dimensions.

2.3. ADVANTAGES OF SPECTRAL RENDERING

Based on the spectral rendering techniques reviewed in section 2.2 and their reported results in prior work, several recurring advantages over RG-based rendering can be identified.

Spectral rendering offers several fundamental advantages over traditional RGB-based rendering pipelines by explicitly modeling light as a function of wavelength. These advantages extend beyond visual accuracy and impact physical correctness, perceptual accuracy, and robustness across varying lighting conditions.

One key advantage of spectral rendering is its ability to accurately simulate wavelength-dependent optical phenomena. Effects such as chromatic dispersion, fluorescence, and polarization occur from interactions that vary across wavelengths and cannot be recreated using RGB-based renderers. Wilkie et al. [5] demonstrate combined rendering of polarization and fluorescence effects that are inherently spectral in nature. Fontas [7] illustrates realistic chromatic dispersion through spectral sampling. Mojzík [9] further shows that fluorescence effects can be captured within a hero wavelength framework, demonstrating the power of spectral pipelines.

Spectral rendering also improves perceptual color accuracy because it inherently avoids the limitations RGB color space have. Yamaguchi et al. [3] explain that RGB representations are fundamentally underdetermined, as multiple distinct spectral distributions can map to identical RGB values, a phenomenon known as metamerism[5]. Peercy [14] and Croisant [35] similarly emphasize that RGB pipelines cannot guarantee consistent color reproduction under different lighting sources. Lastly, Peters [27] demonstrates that spectral renderers produce more stable and physically correct color responses, especially in scenes involving narrowband light sources or complex materials.

Another advantage of spectral rendering is its ability to model interactions between different materials and light sources more accurately. Material reflectance and transmittance properties differ across wavelengths, and spectral rendering preserves this variation throughout the light transport simulation. Peters [25] highlights that operating directly on spectral power distributions enables correct handling of wavelength-dependent absorption and scattering. These are aspects which are typically approximated or ignored in RGB-based systems. For example, in the rendering equation scattering gets handled as a local approximation. It assumes that the light ray hits the surface and immediately reflects from the exact same point. It ignores subsurface scattering[6] and volumetric scattering[7] [6]. Compact spectral representations, such as moment-based approaches [12, 24], allow for these interactions to be captured efficiently without storing full spectra.

Spectral rendering also provides greater robustness under changing lighting conditions. Because spectral pipelines operate independently of specific color primaries, they generalize better across different light sources. Iehl and Péroche [1] demonstrate that perceptually controlled spectral rendering can adapt more effectively to varying lighting scenarios. Perez et al. [28] similarly argue that spectral approaches offer improved consistency compared to RGB renderers that are often tuned for specific light sources.

[5] In spectral rendering, narrowband refers to representing or sampling light over a small wavelength interval rather than over the entire visible spectrum as in RGB rendering. This technique is commonly used in full spectral path tracing [2] but also in hero wavelength sampling [12].
[6] Subsurface scattering refers to the mechanism of light transport where light that penetrates the surface of a translucent object is being scattered. The light enters the material and exits the surface of the material from a point different than the original enter point. [34]
[7] Volume scattering refers to light scattering in volumetric phenomena like clouds, smoke and translucent materials like wax and human skin. [6]

Lastly, spectral rendering forms a strong foundation for predictive rendering systems. Murray et al. [8] show that GPU-oriented spectral rendering can support predictive rendering tasks, while Peters [26] and Perez et al. [28] highlight recent advances that make real-time spectral rendering increasingly feasible.

2.4. CHALLENGES & PERFORMANCE CONSIDERATIONS

Despite its advantages in physical accurate rendering, spectral rendering also introduces several challenges that have limited its widespread adoption in a lot of real-time applications.

The most significant challenge is performance overhead. Working in the spectral domain increases memory usage, sampling complexity, and computation per shading evaluation. Spectral rendering requires sampling and transporting light across many wavelengths. This increases the dimensionality of the rendering problem and leads to higher computational costs. Radziszewski et al. [2] shows that full spectral path tracing significantly increases render times due to the need to evaluate wavelength-dependent material responses, light sources and scattering events at each interaction.

From a sampling perspective, spectral rendering also complicates the Monte Carlo integration. Each additional wavelength dimension increases variation, often requiring more samples to achieve noise levels comparable to RGB renderers. Wilkie et al. [12] highlights that sampling the spectral domain can lead to inefficient convergence, motivating techniques such as hero wavelength sampling to reduce variance while still preserving spectral effects. However, these approaches increase extra complexity in reconstruction and demand carefully designed importance sampling strategies. Additionally, Murray et al. [8] discusses GPU-oriented rendering techniques and emphasizes that naïvely extending RGB pipelines to spectral representations often results in harsh performance penalties.

Another major challenge lies in data representation and memory usage. Spectral rendering requires storing spectral reflectance curves, emission spectra, and wavelength-dependent indices of refraction. Peters [25][26] emphasizes that higher spectral resolution directly increases memory bandwidth usage, which is problematic on GPUs. Memory access patterns are critical for performance on these systems. Moment-based representations [17][32] do address this issue by compressing spectra into a small set of coefficients, but this however causes reconstruction errors and an increased implementation complexity.

Integration into existing rendering pipelines also poses difficulties. Most modern graphics APIs, shading languages like GLSL, and asset pipelines are designed around RGB color spaces. Murray et al. [8] note that extending an RGB-based pipeline to support spectral data requires lots of architectural changes. This includes BRDF evaluations, spectral texture formats, and wavelength aware lighting models. This limits the implementation of spectral rendering in existing production engines that heavily rely on RGB-based tooling and artist workflows.

In real-time context, spectral effects are often not used in scenes. Effects such as dispersion, fluorescence, or polarization are only noticeable under specific conditions. Perez et al. [28] provides a recent overview of techniques aimed at real-time spectral rendering including reduced spectral sampling, hybrid representations and selective evaluation of spectral effects. These approaches suggest that spectral rendering does not need to be applied uniformly across an entire scene to provide perceptual benefits. These selective or hybrid approaches offer a more practical compromise. Similar conclusions are drawn by Peters [27], who demonstrates that many scenes show minimal visual difference between RGB and spectral rendering. When string wavelength-dependent effects are present, visual differences are apparent in the scene.

Finally, validation and perceptual evaluation present additional challenges. Because spectral rendering aims for physical accuracy, evaluating its benefits requires careful comparison against ground truth spectral data or perceptual studies [1][3]. This complicates both development and benchmarking, especially in real-time simulations where performance constrains decide design choices.

3. HYBRID RENDERING METHODS

Hybrid rendering methods seek to balance visual accuracy and computational performance by combining multiple rendering techniques or hardware architectures within a single pipeline. Rather than relying on a single approach such as rasterization rendering techniques, hybrid systems exploit the complementary strengths of different methods to achieve high quality results at important performance critical levels [1][5]. This design philosophy has become increasingly relevant with the emergence of real-time ray tracing hardware and heterogeneous computing platforms.

3.1. HYBRID ALGORITHMIC ARCHITECTURES

One prominent class of hybrid architectures combines rasterization and ray tracing. One the one hand, rasterization is typically used for primary visibility and base shading due to its high output and predictable performance characteristics. On the other hand, ray tracing is applied selectively to compute effects that benefit from physically accurate light transport simulation. For example, shadows, reflections or global illumination.

Heuristic-based hybrid approaches have been explored to determine which parts of a scene should be processed using rasterization versus ray tracing. Walewski et al. [13] investigates hybrid rendering strategies in which heuristics dynamically guide the choice of rendering technique based on scene complexity and expected visual impact. Their results demonstrate that hybrid rasterization/ray tracing pipelines can significantly reduce computational cost while preserving much of the visual accuracy of fully ray traced solutions.

3.2. HYBRID HARDWARE ARCHITECTURES

Hybridization also occurs at the hardware level, where rendering workloads are distributed across heterogeneous computing resources such as CPUs and GPUs. Traditionally, rasterization and simpler shading tasks have been handled on the GPU, whereas the CPUs handle control flow, scheduling, and data preparation. Modern hybrid rendering systems increasingly leverage both processor types concurrently to maximize overall output and resource utilization. [13] Research into CPU/GPU hybrid rendering architectures has shown that distributing tasks according to their computational characteristics improves scalability, particularly for Monte Carlo based rendering algorithms [4].

A concrete example of a hybrid hardware renderer is LuxCoreRender, which supports heterogeneous execution using OpenCL and/or CUDE. This allows for rendering workloads to be distributed across any number of CPUs/GPUs available [15]. LuxCoreRender operates as a full spectral path tracer and demonstrates how hybrid hardware architectures can be used to offset the high computational cost associated with spectral rendering.

3.3. HYBRID SPECTRAL RENDERING IN PRACTICE

Commercial renderers further illustrate the feasibility of hybrid rendering paradigms. OctaneRender is a GPU-accelerated, physically based renderer that employs a hybrid architecture combining spectral light transport with optimized GPU execution [19]. OctaneRender supports spectral rendering for improved physical accuracy while maintaining interactive performance through efficient sampling strategies and hardware acceleration.

Although OctaneRender is primarily targeted at offline and interactive rendering rather than real-time applications, it demonstrates that hybrid spectral approaches are viable when carefully integrated into modern rendering pipelines. These systems provide valuable insight into spectral accuracy and performance constraints can be balanced. [2][8]

CASE STUDY

1. INTRODUCTION

While spectral rendering provides a physically accurate model of light transport and simulates its interactions accurately in a scene, it also enables effects like wavelength-dependent refraction and dispersion.

This case study investigates a hybrid rendering approach that combines RGB and spectral rendering within a single real-time renderer. The goal is to selectively apply spectral light transport only where it provides a perceptual or physical benefit, while maintaining interactive performance comparable to traditional RGB rendering.

In this section, I am trying to answer this research question with the following hypothesis:

"How can a hybrid spectral-RGB rendering pipeline be designed to optimize the trade-off between perceptual color accuracy and real-time rendering performance in modern game engines?"

- **H0**: Implementing selective spectral sampling in a hybrid spectral-RGB pipeline results in a mean color deviation (ΔE) below 5% compared to full spectral rendering, while sustaining a frame rate of at least 60 FPS on modern GPUs.
- **H1**: Limiting spectral computations to reflective and refractive materials yields a higher ratio of visual accuracy to performance cost compared to applying spectral rendering uniformly across all materials.
- **H2**: A post-process spectral-to-RGB conversion stage can reproduce dispersion and fluorescence effects with less than 10% deviation from physically based spectral references, introducing no more than 5 ms per frame on average.

2. SCOPE AND ASSUMPTIONS

2.1. TARGET RENDERING PIPELINES

The primary goal of this study is the implementation and evaluation of a hybrid rendering technique in which both spectral and RGB based light transport are combined within a single real-time renderer. Instead of relying on a single global color representation, the renderer supports multiple pipelines that are different in their representation and evaluation of radiometric quantities during shading. This enables a controlled comparison between traditional RGB rendering, spectral rendering, and a selective hybrid approach.

2.1.1. FULL RGB PIPELINE

The first pipeline is a conventional RGB-based renderer that operates on sRGB radiance values. Material reflectance, emission, and illumination are represented as three-component color vectors, and the rendering equation is evaluated using component-wise operations. This approach closely follows already established real-time rendering practices. While it is highly optimized, it fundamentally lacks the ability to simulate wavelength-dependent phenomena, because all spectral information is collapsed into three fixed color channels.

2.1.2. FULL SPECTRAL PIPELINE

The second pipeline is a spectral renderer that models light transport as a function of wavelengths. In this pipeline, illumination is represented using densely sampled illuminant spectra, while material reflectance is estimated using spectral representations.

Radiance is estimated using a Monte Carlo path tracing method that goes over a discrete set of sampled wavelengths along each light path. Then the radiometric quantities are evaluated separately per wavelength before they are gathered into a RGB response. These calculations enable physically accurate simulation of effects such as dispersion, fluorescence, …

To reduce variance and improve convergence, the estimator also uses importance sampling on both the material BRDF and light sources.

However, the increased dimensionality of the light transport computation increases the computational cost, which limits the feasibility of achieving real-time performance when applied across an entire scene.

2.1.3. HYBRID SPECTRAL-RGB PIPELINE

The third pipeline, which is the focus of this case study, implements a hybrid spectral-RGB renderer that combines both discussed approaches within a single frame. In this pipeline, spectral light transport is evaluated only for scene assets that are expected to produce significant wavelength-dependent effects, while the rest of the scene content is rendered using the RGB pipeline.

By allowing both representations to contribute to a shared radiance buffer, the hybrid pipeline aims to preserve the advantages of spectral rendering for selected materials while retaining the performance characteristics of an RGB-based system for the majority of the scene.

2.2. DEFINITION OF REAL-TIME CONSTRAINTS

For the purpose of this work, real-time performance is defined as maintaining an average frame rate of 60 FPS on modern GPUs. This performance target includes the complete rendering workload, including geometry processing, material evaluation, spectral or RGB shading, accumulation of radiance, tone mapping and post-processing. The performance budget therefore reflects the full cost of the hybrid pipeline rather than isolating individual shader stages.

In addition to frame rate requirements, memory usage is constrained to remain within the limits of available GPU resources without relying on excessive intermediate buffers or off-chip transfers. In particular, the additional data structures required for spectral rendering such as wavelength-sampled illuminant textures and spectral material parameters, must be seamlessly integrated into existing resource layout of the renderer.

To make spectral rendering suitable for interactive applications, the extra computation time it requires is limited. In this study, the spectral part of shading workload is restricted to no more than 5ms per frame, and only for the materials rendered in the spectral domain. This limit ensures that hybrid rendering could be practical in applications such as games, virtual environments, and interactive visualization tools, where responsiveness and smooth frame timings are critical.

2.3. DEFINITION OF HYBRID RENDERING IN THIS WORK

The term hybrid rendering is used in the literature study to describe a wide range of techniques that combine different rendering methods (see header 3). In the context of this case study, hybrid rendering has a more specific meaning: it refers to the integration of a spectral rendering pipeline and an RGB rendering pipeline within a single renderer.

Under this definition, the choice of which pipeline is made at the level of material evaluation rather than at the level of the entire frame. Materials that are expected to exhibit strong wavelength-dependent behavior, such as dispersive glass, prism, or fluorescent surfaces, are calculated using spectral light transport. For these materials, radiance is computed by sampling and integrating across the wavelength domain. The resulting spectral contribution is then converted into a three-value representation only at the point of display.

All remaining materials are shaded using the conventional RGB pipeline. For these surfaces, light transport is evaluated directly in RGB space, avoiding the overhead associated with spectral sampling while still contributing consistently to the final image. The two pipelines don’t run separately. Instead, they work together, adding their radiance values into the same shared high-dynamic-range buffer and go through to the same tone mapping and post-processing stages.

This selective approach allows the renderer to allocate computational effort where it has the greatest perceptual impact. By restricting spectral evaluation to a small subset of materials, the hybrid pipeline aims to maximize visual accuracy for wavelength-sensitive effects while preserving interactive frame rates on current hardware. The case study presented in this paper evaluates whether this balance can be achieved in practice.

3. RENDERER OVERVIEW

For this paper, several existing spectral renderers and rendering frameworks were looked into to assess their suitability as a foundation for implementing selective spectral evaluation. This section outlines the most relevant frameworks that were explored, discusses their limitations in the context of this paper, and motivates the final choice of the base renderer used for the hybrid implementation. The frameworks that will be discussed are the Malia Rendering Framework, Simple Spectral renderer, Epsilon renderer and finally the Path_Tracer renderer.

The evaluation criteria were guided by the goals defined earlier in this work: support for both RGB and spectral rendering, the ability to integrate both approaches within a single pipeline, and a design that is compatible with interactive, GPU-based rendering. While many existing spectral renderers demonstrate impressive physical accuracy, they often target offline use cases or lack the architectural flexibility required for per-material hybrid rendering.

3.1. THE MALIA RENDERING FRAMEWORK [29]

The Malia Rendering Framework (MRF) is an open-source library that includes both an RGB and spectral pipeline. Its design emphasizes physical correctness and extensibility, offering a wide range of spectral representations and material models. From a research perspective, MRF is a highly capable platform and serves as an important reference during the conceptual phase of this project, particularly in its treatment of spectra and its clean separation between color representations.

While this framework is very flexible, its architecture prioritizes offline, high-quality spectral rendering and it is not optimized for real-time interactive scenarios. Its architecture prioritizes flexibility and accuracy over performance, relying on computationally expensive spectral evaluations that are not constrained by strict time budgets. As a result, the framework does not support real-time execution on the GPU.

Furthermore, integrating a selective hybrid approach into MRF would require big architectural changes. The renderer assumes a global color representation per render configuration, making it difficult to switch between RGB and spectral evaluation at shading events. While MRF inspired the conceptual idea of selective spectral evaluation, its offline-oriented design ultimately made it unsuitable as a practical foundation for this project.

3.2. SIMPLE SPECTRAL [10]

Simple spectral, developed by Agatha Mallett and Anders Langlands, was designed as a research platform for evaluating and comparing different spectral rendering techniques. It includes multiple spectral rendering algorithms, including spectral primary decomposition, which is based on their paper “Spectral Primary Decomposition for Rendering with sRGB Reflectance”. The two other algorithms that are included in the rendered are from the papers [Meng et al. 2015] and [Jakob and Hanika 2019]. From a methodological standpoint, Simple Spectral offers a clear implementation of multiple spectral approaches within a single codebase.

Despite these strengths, Simple Spectral was not designed with real-time performance as a target. Its focus lies in algorithmic clarity and correctness rather than GPU efficiency or interactive rendering. The renderer operates in an offline context, with spectral sampling strategies and material evaluations that are too costly to be executed within a tight frame time budget.

3.3. EPSILON [31]

The Epsilon renderer was also considered during the initial exploration phase. Epsilon is a small, C++/OopenCl-based ray tracer and is described as a physically based spectral renderer on its GitHub repository [30]. The renderer uses a fixed wavelength-sampled representation of light and converts spectral radiance to CIE XYZ tristimulus values by using precomputed color matching functions. This design enables physically accurate spectral effects, but the renderer assumes a globally spectral pipeline with a fixed spectral resolution.

Because of the lack of documentation, a code architecture that did not fit the space to implement a selective hybrid renderer and no optional RGB rendering in the framework, Epsilon is less suitable for experimentation in this context and therefore unsuitable for this project.

3.4. PATH_TRACER [18]

The renderer ultimately selected as the basis for this project is the Path_Tracer framework by by Christoph Peters. This renderer is a modern spectral path tracer designed with GPU acceleration in mind. Its modularity and both RGB and spectral pipeline calculation included, made it the ideal starting point for this project.

One of the most significant advantages of Path_Tracer is its explicit support for multiple color models within a rendering framework. The renderer already includes both RGB-based and spectral light transport implementations, allowing direct integration without duplicating the entire pipeline. This made it possible to introduce a hybrid mode by extending existing data structures and shader logic rather than rewriting the renderer from scratch.

The framework's GPU-centric design, implemented using Vulkan and GLSL, provides control over memory layout, resource binding, and synchronization. This level of control is essential for managing the additional data required for spectral rendering, such as illuminant spectra and wavelength sampling, while still meeting real-time performance constraints.

Another key factor in the choice of Path_Tracer is its emphasis on practical performance. The renderer is explicitly designed to balance physical accuracy with computational efficiency, making it well suited for experiments that aim to trade off spectral accuracy against frame time. For these reasons, Path_Tracer was chosen as the base renderer for this paper.

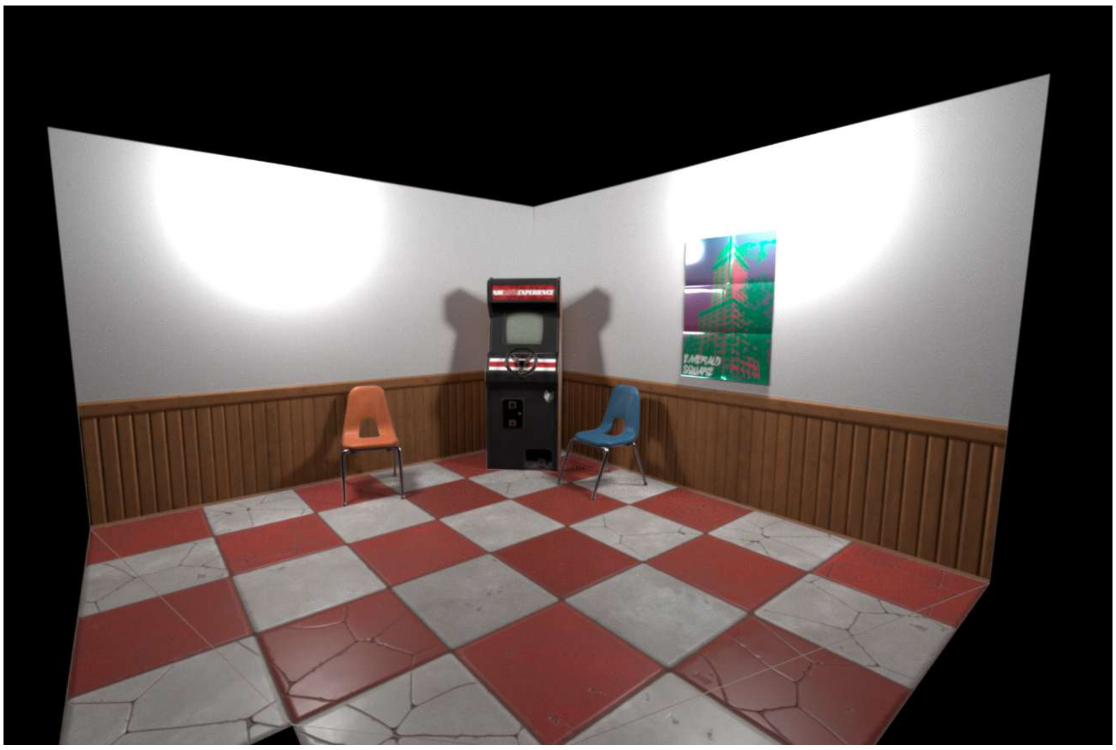


Figure 6: A screenshot, spectral rendered scene, inside the Path_tracer project

4. HYBRID RENDERER DESIGN

The hybrid renderer is designed around the central idea that spectral rendering should only be applied selectively rather than across an entire scene. The motivation for this design came to light (pun intended) from the observation that only a limited subset of materials exhibits perceptually significant wavelength-dependent behavior, while the majority can be rendered adequately using RGB-based light transport. The renderer aims to combine the physical accuracy of spectral rendering with the efficiency of RGB rendering within a single pipeline.

4.1. DESIGN GOALS

A primary design goal is perceptual accuracy. For materials whose appearance is dominated by wavelength-dependent phenomena, such as dispersion in glass or prisms, the renderer seeks to preserve the visual characteristics that are lost in RGB-based approaches.

At the same time, the renderer is limited with strict performance requirements. To remain suitable for interactive applications, the system is designed to sustain frame rates around 60 frames per second by restricting spectral computation to a small subset of materials. RGB shading remains the default path and is used wherever spectral accuracy does not provide a noticeable perceptual benefit. This selective application of spectral rendering is essential for keeping computational cost within real-time bounds.

Flexibility is another central design objective. The renderer supports RGB-only, spectral-only, and hybrid pipelines with a single set of scene assets and shaders. Switching between these modes does not require modification of geometry, textures, or material definitions, but is instead controlled through renderer setting and per-material metadata. This allows direct visual and performance comparisons between different pipelines under identical conditions.

Finally, the hybrid approach is designed to integrate into a modern GPU-based path tracing framework. The renderer is implemented using Vulkan for explicit resource and synchronization control, and GLSL shaders for shading and light transport evaluation. The hybrid logic is embedded directly into the path tracing stage to avoid separate rendering passes or costly pipeline switches.

4.2. HYBRID RENDERING STRATEGY

The hybrid rendering strategy operates by selecting the color evaluation model at the level of individual materials rather than globally. During scene loading, each material is either flagged as either RGB or spectral based on its intended physical behavior. In the current implementation, this classification is derived from material metadata such as naming conventions, where materials associated with dispersive behavior are automatically flagged for spectral evaluation. When the renderer is switched to hybrid mode, this classification determines how each surface interaction is shaded.

The selected color mode for each material is stored in a dedicated GPU buffer that maps material indices to color evaluation modes. This buffer is accessed directly by the path tracing shaders, enabling a branch-free decision at shading time as to whether a surface interaction should follow the RGB or spectral code path. Because the decision is made by material rather than per object or per pass. This makes it so RGB and spectral surfaces can be within the same scene.

During path tracing rays that intersect RGB-classified materials are shaded using standard RGB light transport. Radiance is represented as a three-component vector, and material evaluation follows conventional physically based shading models. For spectral-classified materials, light transport is evaluated by sampling in discrete set of wavelengths-dependent radiance contributions along the path, accounting for spectral emission and reflectance as defined by the illuminant and material models.

Both RGB and spectral contributions are accumulated into the same radiance buffer. Spectral contributions are converted to three-value representations before accumulation, ensuring compatibility with the shared tone mapping and display pipeline. This strategy allows the renderer to leverage spectral accuracy where it matters most, while preserving the performance efficiency of RGB rendering elsewhere.

5. IMPLEMENTATION DETAILS

The hybrid renderer is implemented as an extension of an existing Vulkan-based path tracing framework made by Christoph Peters [18]. The key implementation challenge was enabling per-material selection of the color evaluation model without introducing additional rendering passes or duplicating shader logic.

MATERIAL METADATA UPLOAD

Material classification is performed during scene loading. Once the scene geometry and materials are parsed, an additional buffer is allocated to store per-material metadata indicating the selected color mode. This buffer, referred as the material metadata buffer, contains one integer value per material. It encodes whether the material should be evaluated using RGB or spectral shading. In hybrid mode, this assignment is determined dynamically based on material properties, while in RGB-only or spectral-only modes, all entries are initialized uniformly.

The material metadata buffer is uploaded to the GPU as a storage buffer and bound to the descriptor set used by the path tracing shaders. This ensures that the color mode corresponding to the intersected material is accessible during shading without requiring CPU-side intervention or shader recompilation. The buffer is indexed using the material index provided by the intersection data, allowing constant-time lookup of the shading mode.

SHADER EVALUATION & WAVELENGTH SAMPLING

Within the shader, the material color mode is queried at each surface interaction. If the material is marked as RGB, the shader follows the conventional RGB path tracing logic, evaluating reflectance and illumination using three-component color vectors. If the material is marked with a tag, like “glass” or “prism” for example, the shader switches to a wavelength-sampled evaluation.

In this case, the Monte Carlo tracing is performed across a fixed number of wavelength samples. Each wavelength sample contributes to the accumulated radiance based on the illuminant spectrum and the material’s spectral response.

Spectral contributions are converted to RGB using precomputed illuminant data and an efficient reconstruction method based on compact spectral representation [17]. This conversion is performed entirely on the GPU and is designed to minimize overhead, ensuring that the cost of spectral evaluation remains bounded even in hybrid scenes.

PERFORMANCE OPTIMIZATION

From a performance perspective, the most important optimization is the complete avoidance of spectral computation for RGB materials. Rays interacting with RGB surfaces bypass wavelength sampling, spectral texture lookups, and reconstruction steps entirely. As a result, the additional cost of supporting spectral rendering is pain only for the subset of materials that require it. Post-processing remains unchanged, as both RGB and spectral paths ultimately produce radiance values in a shared buffer that is tone mapped using the same operators.

SCENE PREPARATION AND ASSET CONVERSION PIPELINE

For scene creation and material setup, Blender was used as the main content creation tool. Blender provides an accessible workflow for modeling, lighting, and material authoring, which makes it well suited for creating test scenes. However, the data formats that Blender exports by default are not directly compatible with the data layout and metadata requirements of the Path_Tracer framework. Blender does not export explicit material indices, per-material rendering modes, or GPU-oriented buffer layouts that are required for the hybrid rendering pipeline.

To address this mismatch, the Path_Tracer framework provides a Python-based preprocessing tool that integrates with Blender. This tool is part of the original project and was used as-is in this work. It converts Blender scenes into a set of files that can be loaded directly by the renderer, allowing the authored scenes to be used without manual data conversion.

By relying on the existing Blender-Python export pipeline provided with Path_Tracer, this project could focus on implementing and evaluating the hybrid spectral-RGB renderer itself. The preprocessing tool ensures that scenes created in Blender are imported in a consistent and efficient manner, while still providing the necessary control over material metadata required for selective spectral evaluation.

6. TEST SCENE AND EVALUATION

6.1. VISUALS

The visual results shown in Figure 7 compare the output of the RGB, spectral, and hybrid rendering pipelines under identical scene and lighting conditions. In the RGB configuration, the scene is rendered entirely using a conventional tristimulus color model, resulting in a visually plausible image but lacking wavelength-dependent variation. The fully spectral configuration exhibits subtle but noticeable differences in light distribution and color response, particularly in indirect illumination and high-frequency light interactions, reflecting the influence of wavelength sampling on light transport. The hybrid result closely matches the spectral rendering in regions where spectral evaluation is enabled, while remaining visually consistent with the RGB pipeline elsewhere. This demonstrates that the hybrid renderer successfully preserves perceptually relevant spectral effects without globally applying the more expensive spectral computation.

To validate the correctness of the selective evaluation mechanism, an additional diagnostic experiment was conducted in which RGB-shaded materials were assigned a red debug texture and spectrally shaded materials were assigned a blue debug texture (see Figure 6). In the hybrid configuration, this visualization confirms that spectral evaluation is applied exclusively to the intended materials: the horse statue in the center of the scene, flagged as a dispersive glass material, is rendered using the spectral path, while all surrounding geometry remains on the RGB path. This result verifies that per-material color mode classification is respected during path tracing and that RGB and spectral shading coexist correctly within a single render pass.

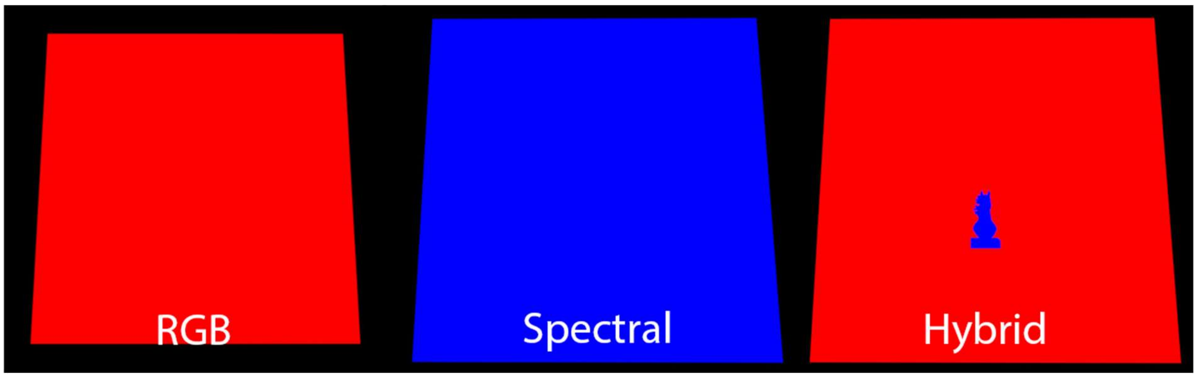


Figure 7: An additional diagnostic experiment to check if the material flags were being recognized

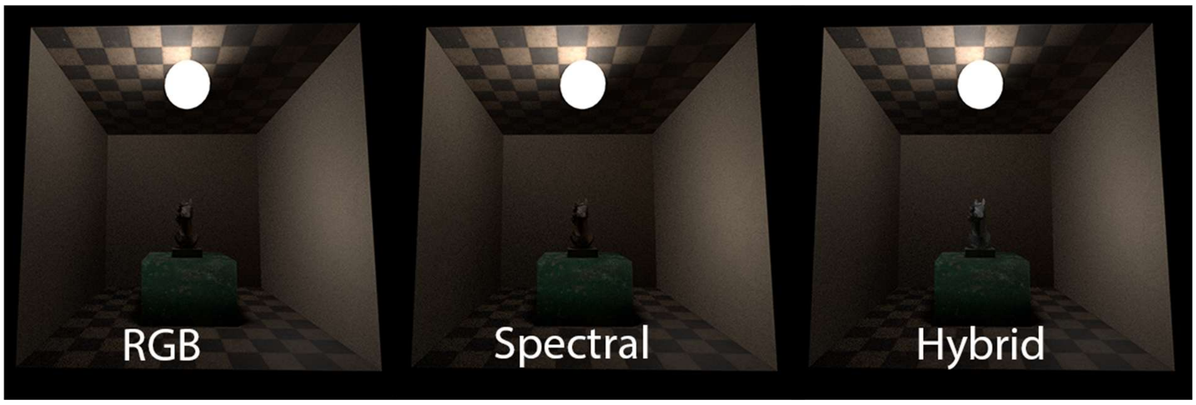


Figure 8: Three identical scenes rendered in RGB, spectral and hybrid pipeline

The grayscale difference images visualize per-pixel color differences relative to the fully spectral reference. Pixel intensity corresponds to the magnitude of the color difference, with darker regions indicating smaller deviations and brighter regions indicating larger differences. These grayscales were visualized using screenshots taken from the renderer, each after a sample period of 2000 frames (See figure 7).

In the RGB-spectral comparison (See Figure 8), slightly elevated color differences are visible across multiple regions of the scene, indicating broad deviations between the RGB pipeline and the spectral reference. These differences are not limited to a specific object and appear in both directly and indirectly lit areas.

In contrast, the hybrid-spectral color differences (See Figure 9) are little more visible than the RGB-spectral color differences. The walls, which are rendered in RGB, have a slightly more elevated color difference while the floor color differences are relatively almost the same. A small but consistent area of increased difference is visible on the spectrally rendered object in the hybrid-spectral color difference photo. This artifact is present in the spectral reference image itself and is therefore passed onto the visualizations.

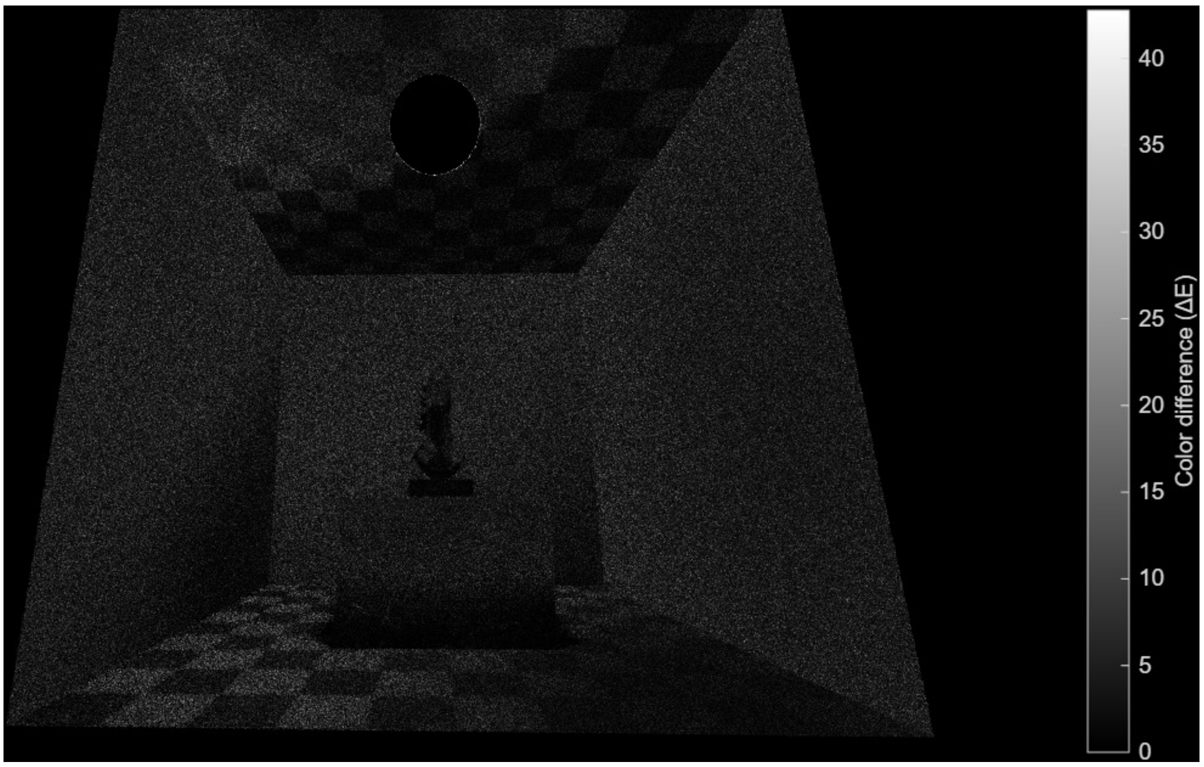


Figure 9: Color difference between RGB and spectral rendered pictures

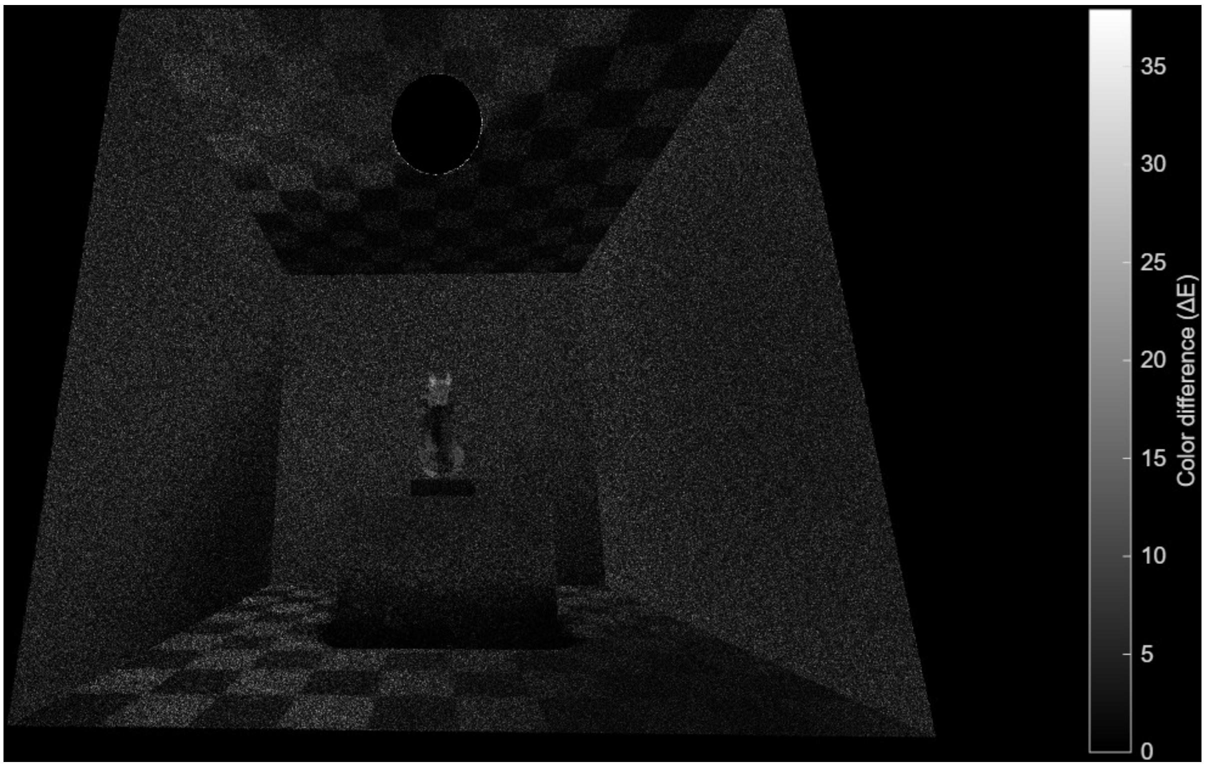


Figure 10: Color difference between Hybrid and spectral rendered pictures

6.2. PERFORMANCE

The performance measurements compare the RGB, spectral and hybrid rendering pipelines using median, mean and percentile-based frame time metrics. Each pipeline was executed ten times, with 300 consecutive frames recorded per run, while the camera moved through the scene from back to front to better approximate a real-time interactive scenario. In the tested scene, the fully spectral pipeline exhibits the highest mean shading time, reflecting the additional computations introduced by wavelengths sampling and spectral material evaluation. The RGB pipeline shows the lowest shading cost (See Figure 9), while the hybrid pipeline falls between the two, with shading times closer to the RGB configuration.

Total frame times across all three pipelines remain similar, with small differences in both mean and median values (See Figures 8 & 9). Post processing times (See Figure 10) are nearly identical across all pipelines, indicating that differences in performance are primarily in the shading stage rather than later stages of the rendering pipeline.

Frame time variability was analyzed using the standard deviation [8]and the 95th percentile[9] (P95) of the total frame time. All three pipelines show low standard deviation values (See Figure 12), suggesting relatively stable frame times during execution. The spectral pipeline exhibits the lowest standard deviation, while RGB and hybrid pipelines show more variation. The measured P95 values (See Figure 11) for all pipelines remain close to their

[8] The standard deviation determines the spread of individual frame times around the mean and provides an indication of frame time stability.
[9] The 95th percentile frame time represents the value below 95% of the recorded frame times fall. It is used to capture occasional performance spikes.

respective mean frame times, indicating that none of the configurations exhibit significant frame time outliers in the tested scene.

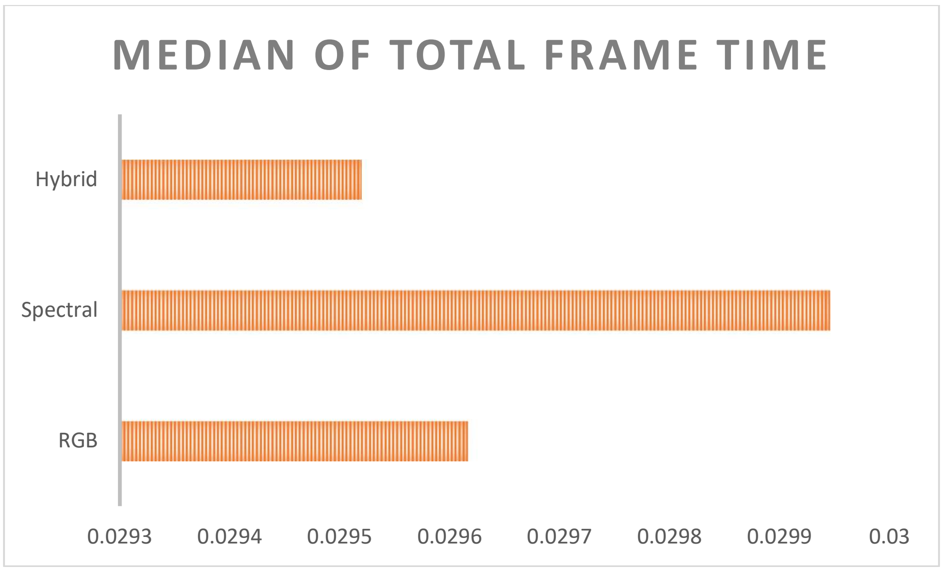


Figure 11: The median of the total frame time in ms per pipeline

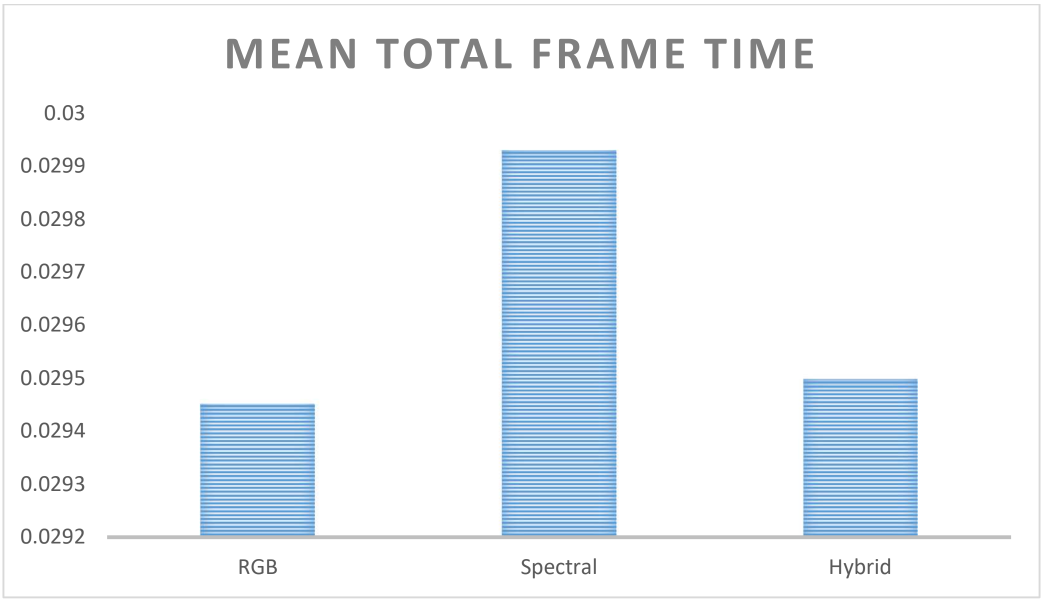


Figure 12: The mean of the total frame time in ms per pipeline

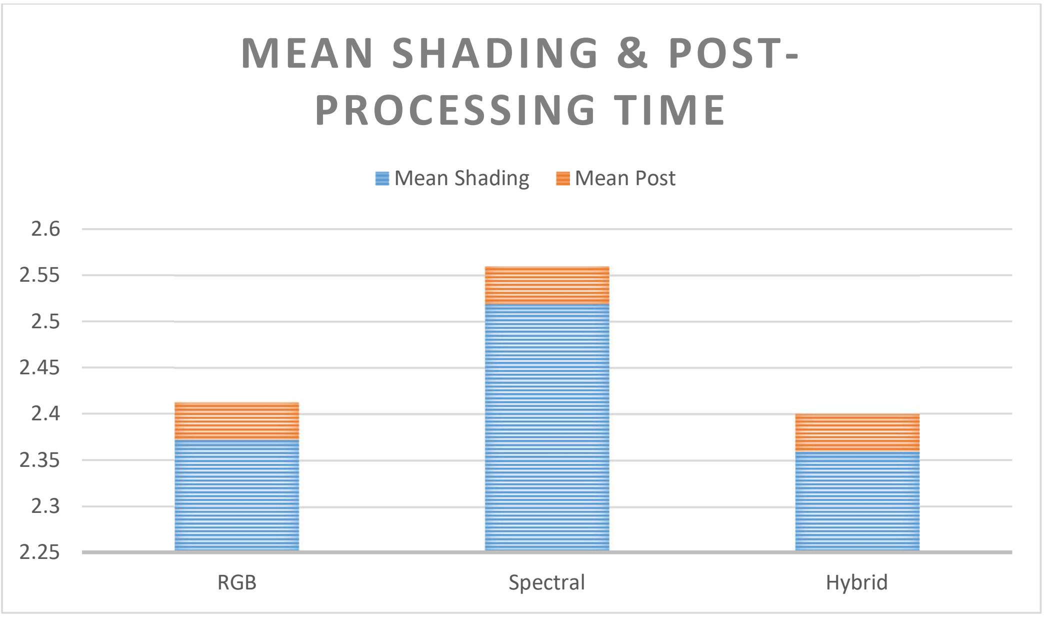


Figure 13: The mean of the total shading and post-processing time in ms per pipeline

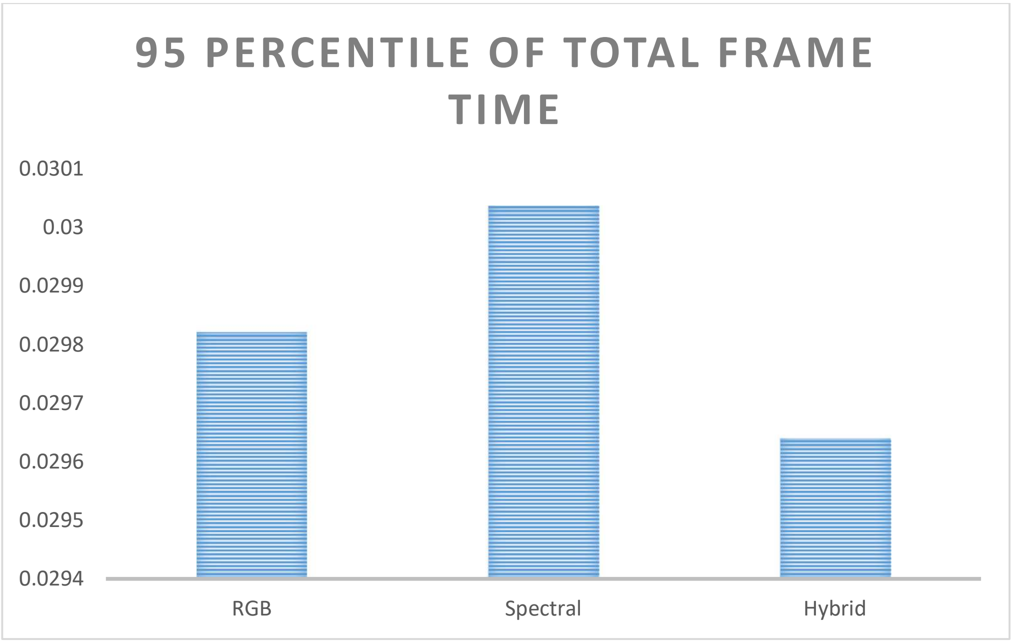


Figure 14: The 95th percentile of the total frame in time in ms per pipeline

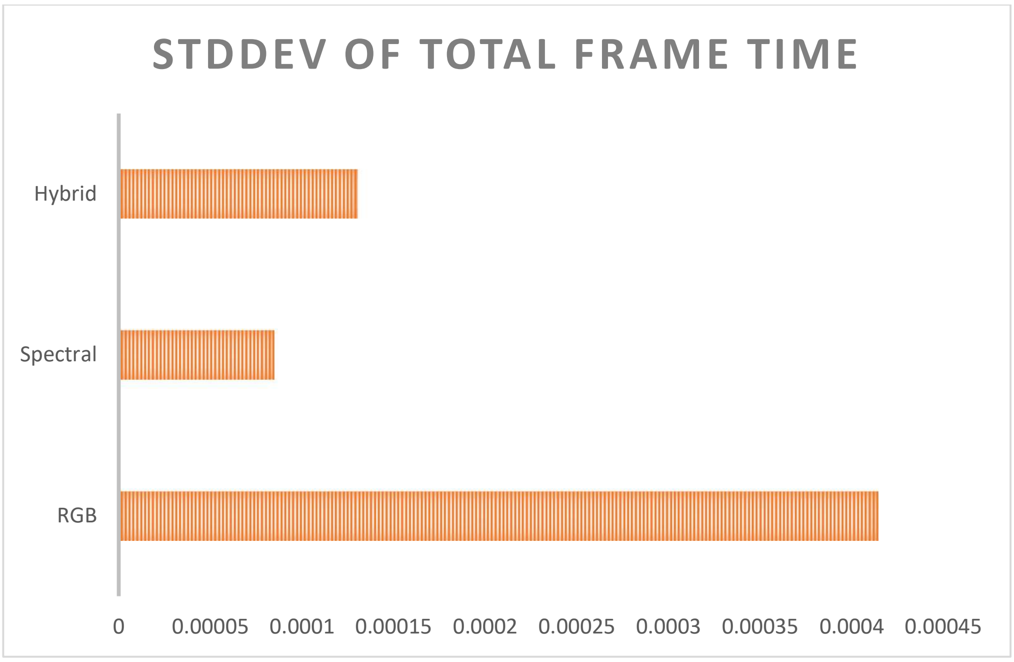


Figure 15: The standard deviation of the total frame time in ms per pipeline

DISCUSSION

This work sets out to investigate whether a hybrid spectral-RGB rendering pipeline could achieve improved perceptual accuracy while remaining compatible with real-time performance constraints. The result obtained in this case study demonstrate that this goal is achievable in practice. By extending an existing GPU-based spectral path tracer and introducing per-material control over color representation, the renderer successfully combines spectral and RGB light transport within a single pipeline.

VISUAL RESULTS

The visual evaluation indicates that the object is rendered spectrally in the hybrid pipeline. In the grayscale, differences in color value between RGB and spectral rendered scenes are clearly visible. The most noticeable differences occur in the glass and prism elements, as well as in shadows and indirect lighting. These observations are consistent with the theoretical claims from the literature review that RGB pipelines struggle to accurately model wavelength-dependent interactions and that spectral renderers can better simulate such effects (Section 1.4 and Section 2.3). Although dispersion and fluorescence were not tested, the observed differences are expected to be even more pronounced in those scenarios.

PERFORMANCE RESULTS

The performance measurements indicate that the hybrid spectral-RGB pipeline achieves lower shading (see Figure 9) costs than the fully spectral pipeline while remaining close to the performance of the RGB configuration. In the tested scene, the fully spectral renderer exhibits the highest average shading time, reflecting the additional cost introduced by wavelength sampling and spectral material evaluation. In contrast, the hybrid pipeline reduces this overhead by restricting spectral computation to a single object, while all other materials follow the RGB path.

Although the difference in total frame time (see Figure 8) between the spectral and hybrid pipelines is relatively small, the mean shading time of the hybrid pipeline is noticeably lower than that of the fully spectral pipeline and closely matches the RGB pipeline. This suggests that selective spectral evaluation can effectively reduce the computational cost of spectral rendering when applied sparingly. Post-processing times remain nearly identical across all three configurations, indicating that the performance differences are primarily attributable to shading rather than to the later stages of the pipeline.

It should be emphasized that these conclusions are based on a limited experimental setup involving a single scene and a single spectrally shaded object in the hybrid configuration. While the results demonstrate that hybrid rendering can reduce spectral overhead in such cases, additional experiments with more complex scenes and a higher proportion of spectral materials are required to generalize these findings.

CONCLUSION

This study set out to answer the research question: **How can a hybrid spectral-RGB rendering pipeline be designed to optimize the trade-off between perceptual color accuracy and real-time rendering performance in modern game engines?** The results support the proposed hypothesis that integrating a selective spectral sampling strategy within a hybrid rendering framework is feasible.

The implementation demonstrates that combining spectral and RGB rendering in a single pipeline can reduce computational overhead while still maintaining visual accuracy in the areas were spectral effects matter most. In the tested scene, selectively applying spectral sampling produced visually more accurate results for those objects without compromising real-time performance. Compared to a fully spectral rendering approach, the hybrid method reduces shading costs and maintained frame rates close to those of the RGB configurations.

Due to the time constraints of this project, the sample amount of this experiment is limited. These results have been tested in only one scene with only one asset marked with a spectral tag and the number of runs was also limited to ten for each pipeline. Thus, the conclusion of this experiment can still change depending on scene complexity, the amount of spectral rendered assets in the hybrid pipeline and more complex wavelength-dependent effects.

FUTURE WORK

These results, however, provide a foundation for future work, in which a broader range of reflective and refractive materials, as well as more complex scenes, could be explored to further validate the scalability and advantages of a hybrid approach. This framework presented in this paper provides a structured basis for extending the current findings and also supports further research into hybrid spectral-RGB rendering techniques. Moreover, this work serves as a suitable foundation for future research at the master's level.

While the hybrid renderer presented in this study demonstrates the viability of selective spectral evaluation, several directions remain open for further research and improvement. One natural extension would be to implement and compare additional spectral rendering algorithms, such as hero wavelength sampling or moment-based spectral representations and evaluate how they perform within the same hybrid framework. This would provide deeper insight into the trade-offs between different spectral models in real-time contexts.

Another promising direction lies in expanding the scope of physically accurate simulation. The current implementation focuses on material-level spectral effects, but future work could explore more extreme wavelength-dependent phenomena, such as relativistic or astrophysical effects. Although these fall outside the constraints of real-time rendering, they could serve as valuable test cases for evaluating the expressive power of spectral pipelines.

The current renderer converts spectral results to RGB before accumulation, effectively treating spectral evaluation as an intermediate step. A more comprehensive approach would involve maintaining fully spectral radiance throughout the pipeline and performing RGB conversion only at the final display stage. While more costly, this would eliminate residual approximations introduced by early conversion and provide a clearer comparison with fully spectral renderers.

Another idea is to improve the control when spectral evaluation is enabled to a heuristic-based control. In the present implementation, materials are classified statically, but future systems could make this decision dynamically. For example, spectral evaluation could be activated only after the first dispersive or refractive bounce or based on estimated perceptual impact. Such heuristics could further reduce the cost of spectral rendering while preserving visual quality, bringing hybrid rendering closer to practical deployment in real-time engines.

Finally, the last additional area for future work I will talk about concerns noise reduction. Due to the random distribution that happens during Monte Carlo path tracing, both RGB and spectral pipelines have visible noise at low sample counts and especially when the camera moves around the scene. In the current implementation, noise reduction is achieved only through increasing the sample counts, but this directly impacts performance and limits real-time applicability. Future work could explore the integration of denoising techniques tailored to hybrid rendering, such as spatial filtering or machine learning based denoisers. Reducing perceptual noise at runtime would improve visual stability and user experience, further strengthening the case for hybrid spectral-RGB rendering in interactive applications.

CRITICAL REFLECTION

This project originated from my passion in graphics programming aspiration to explore specialized domains within graphics programming. Upon discovering spectral rendering, I recognized it as a challenging topic that aligned with my ambition to produce high level technical work.

One of the primary challenges I encountered was the vastness of the field. As a person who likes to read a lot, I initially found myself diving deep into active research areas that, while very interesting, were not always central to my specific research goals. This process, however, taught me a valuable academic lesson. I learned to distinguish between interesting supplementary information and core information necessary for my paper.

By eventually setting strict boundaries on the scope of my reading, I managed to balance my curiosity with the practical constraints of this course. This experience has not only deepened my understanding of spectral rendering but also refined my ability to manage more complex research projects. I would like to hopefully continue this paper by applying to the master's degree in Breda and using this paper as a reference for my topic.

Even though I developed a complete project roadmap, adhering to it didn't go as planned. The planning consisted of the bi-weekly deadlines and my own internal deadlines. I sometimes underestimated the amount of time that would go into tasks like writing my theoretical framework or finding an appropriate framework to implement my hybrid pipeline and conduct the experiments in. Because of this experience, I did improve this semester time management wise and know how to better approach plannings for academic research.

One of the key decisions that helped me achieve the results in this paper was choosing the right framework for the hybrid implementation. Selecting an appropriate framework is crucial to my work because it affects the feasibility of the experiments, the clarity of the pipeline and the over quality of the results. By choosing a framework that supported both hybrid rendering and efficient experimentation, I was able to focus on the core research question. The search for a suitable framework also had a negative impact on my planning. I didn't expect to go over so many frameworks, and I also didn't consider the testing and research that goes into selecting a fitting framework for a paper. Because of the extra time I spend researching different frameworks, I got behind on my planning.

REFERENCES

[1]

J. Claude Iehl and B. Péroche, “An Adaptive Spectral Rendering with a Perceptual Control,” *Computer Graphics Forum*, vol. 19, no. 3, pp. 291–300, 2000, doi: 10.1111/1467-8659.00421.

[2]

M. Radziszewski, K. Boryczko, and W. Alda, “An Improved Technique for Full Spectral Rendering”.

[3]

M. Yamaguchi, H. Haneishi, and N. Ohyama, “Beyond Red–Green–Blue (RGB): Spectrum-Based Color Imaging Technology,” *jist*, vol. 52, no. 1, pp. 10201-1-10201–15, Jan. 2008, doi: 10.2352/J.ImagingSci.Technol.(2008)52:1(010201).

[4]

A. Pajot, L. Barthe, M. Paulin, and P. Poulin, “Combinatorial Bidirectional Path-Tracing for Efficient Hybrid CPU/GPU Rendering,” *Computer Graphics Forum*, vol. 30, no. 2, pp. 315–324, 2011, doi: 10.1111/j.1467-8659.2011.01863.x.

[5]

A. Wilkie, R. F. Tobler, and W. Purgathofer, “Combined Rendering of Polarization and Fluorescence Effects,” in *Rendering Techniques 2001*, S. J. Gortler and K. Myszkowski, Eds., in Eurographics. , Vienna: Springer Vienna, 2001, pp. 197–204. doi: 10.1007/978-3-7091-6242-2_18.

[6]

S. Rotenberg, “CSE 168: Rendering Algorithms,” p. 38, Spring 2017.

[7]

C. Fontas, “Dispersion with Spectral Rendering.” Accessed: Oct. 05, 2025. [Online]. Available: https://chrisfontas.wordpress.com/2014/06/20/dispersion-with-spectral-rendering/

[8]

D. Murray, A. Fichet, and R. Pacanowski, “Efficient Spectral Rendering on the GPU for Predictive Rendering,” in *Ray Tracing Gems II*, A. Marrs, P. Shirley, and I. Wald, Eds., Berkeley, CA: Apress, 2021, pp. 673–698. doi: 10.1007/978-1-4842-7185-8_42.

[9]

M. Mojzík, “Fluorescence Computations in a Hero Wavelength Renderer,” Jan. 2018, Accessed: Oct. 09, 2025. [Online]. Available: https://dspace.cuni.cz/handle/20.500.11956/94715

[10]

A. Mallett, geometrian/simple-spectral. (Dec. 02, 2025). C++. Accessed: Jan. 16, 2026. [Online]. Available: https://github.com/geometrian/simple-spectral

[11]

F. Delombaerde and P.-J. Vandenberghe, “GP1_Raytracing_Lighting.”

[12]

A. Wilkie, S. Nawaz, M. Droske, A. Weidlich, and J. Hanika, “Hero Wavelength Spectral Sampling,” Computer Graphics Forum, vol. 33, no. 4, pp. 123–131, 2014, doi: 10.1111/cgf.12419.

[13]

P. Walewski, T. Gałaj, and D. Szajerman, “Heuristic based real-time hybrid rendering with the use of rasterization and ray tracing method,” Open Physics, vol. 17, no. 1, pp. 527–544, Jan. 2019, doi: 10.1515/phys-2019-0055.

[14]

M. S. Peercy, “Linear color representations for full spectral rendering,” in Proceedings of the 20th annual conference on Computer graphics and interactive techniques, in SIGGRAPH ’93. New York, NY, USA: Association for Computing Machinery, Sep. 1993, pp. 191–198. doi: 10.1145/166117.166142.

[15]

“LuxCoreRender – Open Source Physically Based Renderer.” Accessed: Jan. 04, 2026. [Online]. Available: https://luxcorerender.org/

[16]

“Metamerism (color),” Wikipedia. Oct. 11, 2025. Accessed: Jan. 18, 2026. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Metamerism_(color)&oldid=1316343094

[17]

L. Tódová, A. Wilkie, and L. Fascione, “Moment-based Constrained Spectral Uplifting,” Eurographics Symposium on Rendering - DL-only Track, p. 10 pages, 2021, doi: 10.2312/SR.20211304.

[18]

C. Peters, “MomentsInGraphics/path_tracer at spectral,” GitHub. Accessed: Dec. 31, 2025. [Online]. Available: https://github.com/MomentsInGraphics/path_tracer

[19]

“OctaneRender® Spectral GPU Path Tracer,” Gaussian Spectrum. Accessed: Jan. 04, 2026. [Online]. Available: https://gaussianspectrum.co/octanerender/introduction

[20]

G. Ward and E. Eydelberg-Vileshin, “Picture Perfect RGB Rendering Using Spectral Prefiltering and Sharp Color Primaries.,” in ResearchGate, Jan. 2002. Accessed: Oct. 17, 2025. [Online]. Available: https://www.researchgate.net/publication/220853008_Picture_Perfect_RGB_Rendering_Using_Spectral_Prefiltering_and_Sharp_Color_Primaries

[21]

C. Peters, “Radiometry, part 1: I got it backwards,” Moments in Graphics. Accessed: Dec. 31, 2025. [Online]. Available: https://momentsingraphics.de/Radiometry1Backwards.html

[22]

Peters, Christoph, “Radiometry, part 2: Spectra and photometry.” Accessed: Dec. 31, 2025. [Online]. Available: https://momentsingraphics.de/Radiometry2Photometry.html

[23]

“Spectral power distribution,” Wikipedia. Sep. 11, 2024. Accessed: Jan. 18, 2026. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Spectral_power_distribution&oldid=1245235568

[24]

A. Mallett and C. Yuksel, “Spectral Primary Decomposition,” 2019, [Online]. Available: https://diglib.eg.org/items/bbffa865-e99c-4c1f-bd33-70102dc8af78

[25]

C. Peters, “Spectral rendering, part 1: Spectra.” Accessed: Dec. 31, 2025. [Online]. Available: https://momentsingraphics.de/SpectralRendering1Spectra.html

[26]

C. Peters, “Spectral rendering, part 2: Real-time rendering.” Accessed: Dec. 31, 2025. [Online]. Available: https://momentsingraphics.de/SpectralRendering2Rendering.html

[27]

C. Peters, “Spectral rendering, part 3: Spectral vs. RGB.” Accessed: Dec. 31, 2025. [Online]. Available: https://momentsingraphics.de/SpectralRendering3Results.html

[28]

P. J. Perez, N. Monzon, and A. Muñoz, “Techniques for Real-Time Spectral Rendering,” . ISSN, vol. 13, 2025, [Online]. Available: https://www.researchgate.net/publication/394235001_Techniques_for_Real_Time_Spectral_RenderingTecnicas_para_renderizado_espectral_en_tiempo_real

[29]

“The Malia Rendering.” Accessed: Oct. 19, 2025. [Online]. Available: https://mrf-devteam.gitlab.io/mrf/main.md.html

[30]

J. T. Kajiya, “The rendering equation,” SIGGRAPH Comput. Graph., vol. 20, no. 4, pp. 143–150, Aug. 1986, doi: 10.1145/15886.15902

[31]

T. Bénéteau, TomCrypto/epsilon. (Sep. 04, 2025). C++. Accessed: Jan. 16, 2026. [Online]. Available: https://github.com/TomCrypto/epsilon

[32]

C. Peters, S. Merzbach, J. Hanika, and C. Dachsbacher, “Using moments to represent bounded signals for spectral rendering.” Accessed: Dec. 31, 2025. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/3306346.3322964

[33]

“Visible Light - NASA Science.” Accessed: Jan. 04, 2026. [Online]. Available: https://science.nasa.gov/ems/09_visiblelight/

[34]

“Volume Scattering Processes.” Accessed: Jan. 03, 2026. [Online]. Available: https://pbr-book.org/3ed-2018/Volume_Scattering/Volume_Scattering_Processes

[35]

J. Croisant, “When RGB is Not Enough · croisant.net.” Accessed: Oct. 05, 2025. [Online]. Available: https://croisant.net/blog/2007-09-05-when-rgb-is-not-enough/

ACKNOWLEDGEMENTS

I would like to thank my supervisor, **Koen Samyn**, for his guidance, patience, and continuous feedback throughout this project. His support helped me navigate both the technical challenges and the academic direction of this paper.

I also want to thank my teacher, **Matthieu Delaere**, for helping me with all spectral rendering related question. The additional sources he provided, helped me a great deal with this project.

I am deeply grateful to my family and parents for their constant support and encouragement. Their genuine interest in my work, even when the technical details were difficult to explain or understand, meant more to me than they might realize and helped me stay motivated during challenging moments.

Finally, I want to thank my dear friends **Daan Demaecker**, **Ines Stephanie Bouwen**, and **Effy Peralta Camacho**. They were always there to listen, to encourage me, and to help whenever they could. Whether through technical discussions or simply by offering support when I needed it most. This work would not have been possible without them.

APPENDICES

ABBREVIATIONS

ΔE: Color Deviation

API: Application Programming Interface

BRDF: Bidirectional Reflectance Distribution Function

CPU: Central Processing Unit

FPS: Frames Per Second

GLSL: OpenGL Shading Language

GPU: Graphics Processing Unit

Ms: Milliseconds

Nm: Nanometers

P95: 95th percentile

PBR: Physically Based Rendering

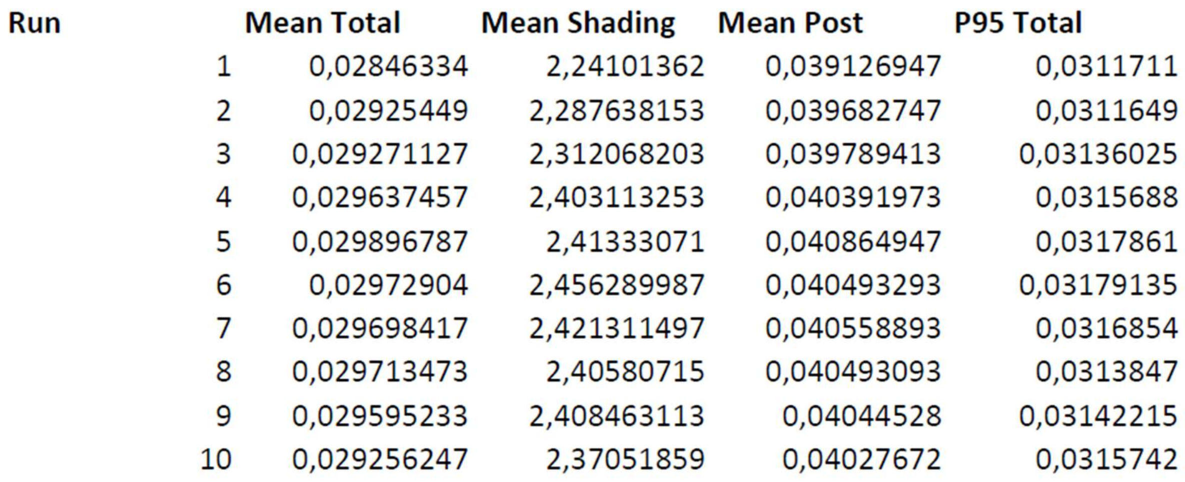
RGB: Red-Green-Blue

APPENDIX I – SOURCE CODE

Underneath is the link to the GitHub page with the forked project. It has the build and run instructions listed in the ReadMe. The hybrid renderer is situated in the branch called hybrid, not the main.

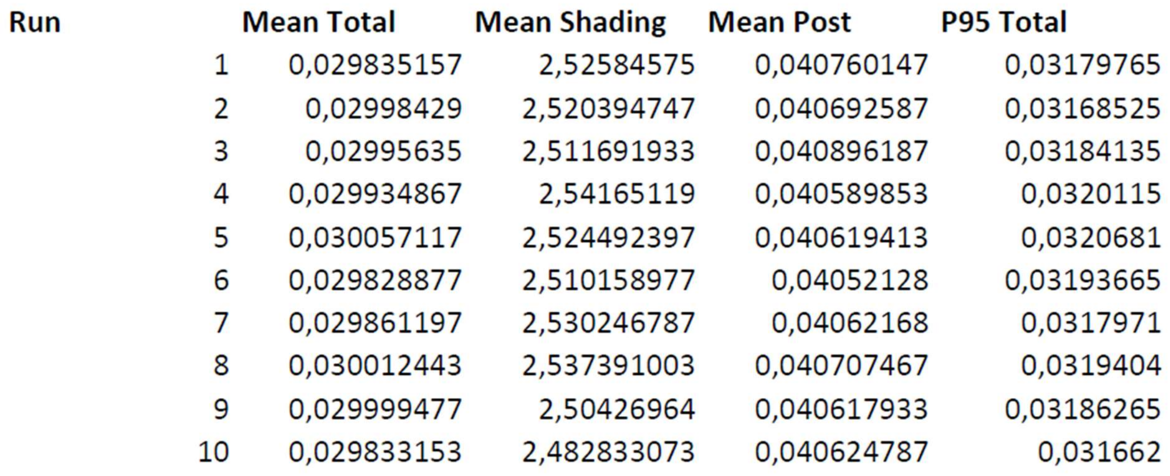
https://github.com/BobsIsHere/HybridSpectralRenderer

APPENDIX II – PERFORMANCE SHEETS



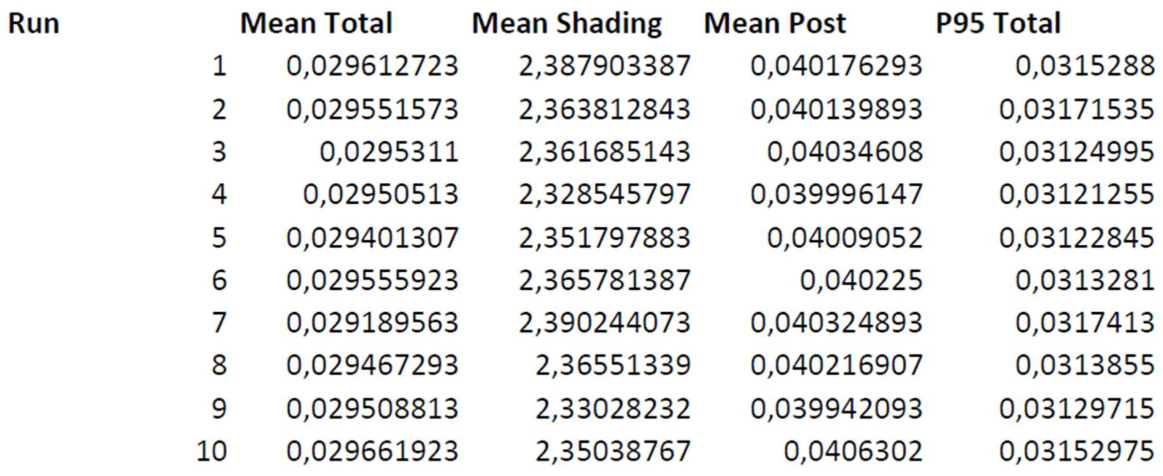
Run	Mean Total	Mean Shading	Mean Post	P95 Total
1	0,02846334	2,24101362	0,039126947	0,0311711
2	0,02925449	2,287638153	0,039682747	0,0311649
3	0,029271127	2,312068203	0,039789413	0,03136025
4	0,029637457	2,403113253	0,040391973	0,0315688
5	0,029896787	2,41333071	0,040864947	0,0317861
6	0,02972904	2,456289987	0,040493293	0,03179135
7	0,029698417	2,421311497	0,040558893	0,0316854
8	0,029713473	2,40580715	0,040493093	0,0313847
9	0,029595233	2,408463113	0,04044528	0,03142215
10	0,029256247	2,37051859	0,04027672	0,0315742

Figure 16: All measurements taken in the RGB pipeline, which consists of 10 runs with each run capturing 300 frames. Each run has its calculated total frame time mean, total shading time mean, total post-processing mean and its P95



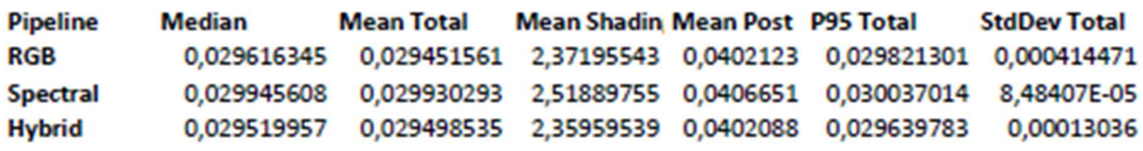
Run	Mean Total	Mean Shading	Mean Post	P95 Total
1	0,029835157	2,52584575	0,040760147	0,03179765
2	0,02998429	2,520394747	0,040692587	0,03168525
3	0,02995635	2,511691933	0,040896187	0,03184135
4	0,029934867	2,54165119	0,040589853	0,0320115
5	0,030057117	2,524492397	0,040619413	0,0320681
6	0,029828877	2,510158977	0,04052128	0,03193665
7	0,029861197	2,530246787	0,04062168	0,0317971
8	0,030012443	2,537391003	0,040707467	0,0319404
9	0,029999477	2,50426964	0,040617933	0,03186265
10	0,029833153	2,482833073	0,040624787	0,031662

Figure 17: All measurements taken in the Spectral pipeline, which consists of 10 runs with each run capturing 300 frames. Each run has its calculated total frame time mean, total shading time mean, total post-processing mean and its P95



Run	Mean Total	Mean Shading	Mean Post	P95 Total
1	0,029612723	2,387903387	0,040176293	0,0315288
2	0,029551573	2,363812843	0,040139893	0,03171535
3	0,0295311	2,361685143	0,04034608	0,03124995
4	0,02950513	2,328545797	0,039996147	0,03121255
5	0,029401307	2,351797883	0,04009052	0,03122845
6	0,029555923	2,365781387	0,040225	0,0313281
7	0,029189563	2,390244073	0,040324893	0,0317413
8	0,029467293	2,36551339	0,040216907	0,0313855
9	0,029508813	2,33028232	0,039942093	0,03129715
10	0,029661923	2,35038767	0,0406302	0,03152975

Figure 18: All measurements taken in the Hybrid pipeline, which consists of 10 runs with each run capturing 300 frames. Each run has its calculated total frame time mean, total shading time mean, total post-processing mean and its P95



Pipeline	Median	Mean Total	Mean Shadin	Mean Post	P95 Total	StdDev Total
RGB	0,029616345	0,029451561	2,37195543	0,0402123	0,029821301	0,000414471
Spectral	0,029945608	0,029930293	2,51889755	0,0406651	0,030037014	8,48407E-05
Hybrid	0,029519957	0,029498535	2,35959539	0,0402088	0,029639783	0,00013036

Figure 19: For each pipeline, the calculated median, total frame time mean, total shading time mean, total post-processing mean, its P95 and standard deviation