Master Thesis

Rendering Granular Materials at the Mesoscopic Scale

Authors

Naglis Naslenas - s223312

January 31, 2025

Contents

Abstract

This master thesis explores the rendering of granular materials at a mesoscopic level in real-time. The goal of the project is to develop a real-time rendering solution that represents granular aggregate as individual particles, focusing on accurately representing the appearance of a single piece as well as making considerations for the medium as a whole. The work explores initial simple diffuse reflection model to establish equal conditions for the comparisons between the real-time and offline renderer, implementation of physically accurate specular reflection model and three methods to represent individual grains are explored and implemented focusing on light interactions within the grain object by leveraging the concept of Bidirectional Surface Scattering Distribution Function (BSSRDF) to account for these complex light interactions. These methods were then compared to the results from an offline renderer PBRT. Findings from this project suggest that the three proposed methods for representing individual grains in the real-time could be used to represent materials of different properties and appearances.

1 Introduction

Granular materials are a materials composed of a large number of individual particles or grains. This includes a variety of aggregates such as snow, sand, powders and even soil. In computer graphics, these types of materials are usually presented at a macroscopic level, where the individual small bits are not visible and instead the material is represented as a homogeneous medium. Visualizing granular materials at a mesoscopic level, where these individual particles are visible, is a challenging task due to complex light interactions within each grain and between them and the surrounding environment. This thesis in collaboration with *Playdead*, a Danish game development studio, aims to explore the rendering of granular materials at mesoscopic level in real-time to further enhance the realism and visual quality of video game graphics.

1.1 Motivation

The incentive behind this project a need for realistic rendering of granular materials in industries such movie production or video games. While there are many modern rendering techniques capable of producing very realistic images, granular materials are still a challenge due to complex light interactions not only within the particles themselves, but also between the particles and the surrounding environment. This is crucial for achieving a realistic look of such materials in virtual environments. Offline renderers are capable of producing realistic solutions to this problem, but it is heavily constrained by the computational demands that come with it. Concessions as precomputations and clusterring must be made in order to achieve a reasonable rendering time along with a balance in quality [1]. Current real-time rendering techniques rely on methods leveraging texture, bump and normal maps to create an appearance of granular aggregates, but are limited in their ability to capture complex light scattering properties of individual grains. Inspired by the research efforts tailored towards offline rendering of granular materials [2], the project aims to develop and implement a real-time rendering solution capable of approximating the results of path tracing for granular materials at the mesoscopic scale, while maintaining real-time performance.

1.2 Problem statement

The main goal of this project, as mentioned, is to explore the rendering of granular materials at a mesoscopic level, focusing on the accurately representing the appearance of individual particles as well as making considerations for the aggregate as a whole accounting for irregular shapes and sizes, various levels of roughness and reflectivity influencing the light scattering and absorption affecting the overall appearance of the material on a larger scale. This will be achieved by implementing a real-time rendering rendering solution that will be referenced with a state-of-the-art offline renderer, *PBRT* [3].The real-time solution, will be implemented using OpenGL and GLSL, and will be compared to the reference renderer in terms of visual quality and assessed for scalability.

2 Related work

In this section we will discuss the related work in the field of granular materials rendering, focusing on the crucial aspects of light interactions and scattering within the material.

As touched upon, granular materials are a challenging subject in computer graphics due to the complex light interactions requiring approximations and simplifications to even be feasible in offline rendering. Research in this field has been focused on developing efficient multi-scale models for rendering granular materials.

The work by Meng et al. [2] explored approaches to find more efficient ways to render different scales of grain based on the scale - explicit path tracing for the visible grains with further approximating granular material as a medium for volumetric path tracing and lastly at the discernable scale - diffuse-based approximation. They are further separating the light interactions within and between the grains to simplify the rendering process.

Building on this foundation Müller et al. [1] introduced using spherical proxies to approximate grains. This technique allows simplify the rendering process reducing its dimensionality and computational complexity, further speeding up the rendering process.

Rendering hair presents similar challenges to rendering granular materials, as it requires accounting for light interactions both within individual strands and between multiple strands and their environment. Zinke et al. [4] addressed these challenges by developing dual scattering to effectively manage light interactions in hair rendering.

Internal light interactions within the grains are also a crucial aspect of rendering granular materials. This established by the work of Jensen et al. [5] introducing the concept of Bidirectional Surface Scattering Distribution Function (BSSRDF) to account for the light interactions within the material. this allows to represent subsurface scattering - crucial light interaction event for granular aggregates as internal scattering and light transport within the material affects the overall appearance of the material. Dal Corso [6] and Frisvad et al. [7] extended this concept to real-time rendering, making it feasible for interactive applications.

3 OpenGL setup

While any modern graphics API could be used for the implementation of the real-time solution, it was chosen to use OpenGL due to its wide adoption and support across different platforms. This was done on the latest (device supported) version of OpenGL 4.1. Window management was handled using GLFW, an open-source multi platform library allowing for the creation of windows, OpenGL contexts and input handling[8] and then compiled using a *MakeFile* [9].

In addition *GLAD* library was used to load the OpenGL functions, generating the necessary header files and source files for the OpenGL functions [10].

4 PBRT

PBRT or Physically Based Rendering Toolkit is used for this project as a reference renderer. It is a physically based rendering system delivering photorealistic rendering using Monte Carlo path tracing. It's biggest strength is extensively detail documentation ranging from the foundational principles of rendering to the implementation of complex rendering techniques. While it is technically possible to extend its capabilities further, it was used as is for the purposes of this project.

To utilize PBRT the scene description files are written in a custom file format [11], which can then be used to render the scene using the PBRT executable. These scene files are written in human readable form but it was necessary to delve into the documentation to comprehend the syntax and explore different available options to get desired results and other intricacies of the renderer. For example it should be noted that PBRT uses a left handed coordinate system, where z points away from the camera, important for the correct orientation of the scene and the objects within it. While simple shapes can be described from the file format, more complex shapes needed to be imported using external tools (Section 5.1.1).

PBRT supports an array of different materials that can be applied to objects in the scene. For this project, and the later comparisons with the real-time renderer, the materials that were used were **diffuse**, **conductor** and **subsurface**. The description of the properties of these materials will be discussed in the following sections.

5 Scene Setup

This section aims to describe the process of setting up a an equivalent scene with an object, light, and camera for both OpenGL and PRBT-v4. For successful comparison of the results from the renderers, it was necessary to align the properties of the scene in both environments.

5.1 Object

As a simple object on which the shading can be more easily observed, a sphere was chosen as its geometry is close to that of the sand grain in real life. The sphere was created using Blender, with a diameter of 4.8 units and of varying vertex count for testing purposes. The sphere was then exported as an .obj file, so that it could be loaded into the OpenGL and PRBT-v4 environments.

A grain model was also created mimicking a sand grain. This was also modeled using Blender using an image of a sand grain as a reference. The original grain consists of 5.24 million vertices but a simplified version of the grain was used for the purpose of this project (Figure 1).

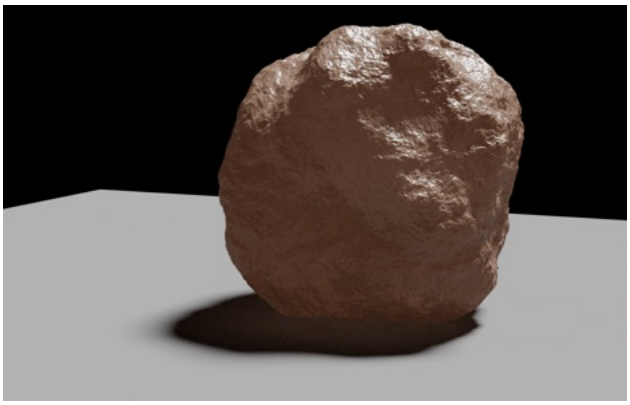


Figure 1: Grain model created in Blender

5.1.1 Exporting object to PBRT

It is important to note that the .obj file exported from Blender must be converted to a .pbrt file to be used in the PRBT-v4 environment. To do this convertion, Open Asset Import Library (assimp) [12] was used which is capable of loading various 3D file formats to a shared format, supporting numerous different file formats. In this case, this library was used to load the contents of the .obj file and then write the equivalent data to a file with the .pbrt extension. The resulting file can then be loaded into the PRBT-v4 environment as a scene object or by removing the *world* information, it can be used as an object in an existing PBRT scene.

5.1.2 Loading object in OpenGL

To load the object in OpenGL, the .obj file is also parsed using assimp library with a wrapper class Model availalbe from *learnopengl* [13] which allows for easy loading and further use of data of the object. The object is then ready to be rendered in the scene using the OpenGL rendering pipeline.

5.2 Light

Given the size of granular materials in a scene, the light source is at a distance far enough to be considered a directional light source and for an object of such scale there is not a significant difference in the lighting of the scene when using a point light source or a directional light source. Therefore, two directional lights were used for the scene - a front light at a direction of *[1, 1 ,1]* and a back-light causing grazing angle reflections at *[-1, -1, -1.6]* [14]. Light intensity was set to *20.0* and *10.0* respectively, emitting white light. This setup was used for both OpenGL and PBRT-v4 environments.

5.3 Camera

A standard pinhole camera was used for OpenGL with an equivalent PBRT camera setup with a lens radius of 0 to disable depth of field and lens distortion effects, focal distance set to infinity and a field of view of 45 degrees. The camera was placed at a distance of 6.5 units from the center of the sphere looking at it's center.

5.4 Image Inspection using TEV

Renderings of the images were inspected using the TEV tool, which facilitates detailed inspection of individual pixel channel values. This tool is invaluable for debugging rendering issues, adjusting exposure and gamma settings, and analyzing image histograms to understand pixel value distributions. Additionally, TEV provides critical information on the minimum, mean, and maximum pixel values, further adding to the utility of the tool.

Besides the inspection of a single image, TEV tool also allows to compare two images by selecting a reference image and then comparing it with another. This functionality is highly useful for observing incremental changes when improving the rendering process, modifying object's material properties or comparing the outputs from different rendering engines. Comparing two images allows to visually inspect the differences between the two images as an error (E), absolute error (AE), relative absolute error (RAE) or relative squared error (RSE) values which can then be used to quantify the differences.

For the results from the project RSE values will be used to quantify the differences between the images as it penalizes large differences more than small differences. The RSE in TEV is calculated as follows:

$$RSE = \sqrt{\frac{(i-r)^2}{(r^2+0.01)}} \tag{1}$$

where i is the pixel value of the image being compared and r is the pixel value of the reference image. The RSE value is then normalized by the reference pixel value squared and a small constant value of 0.01 to avoid division by zero.

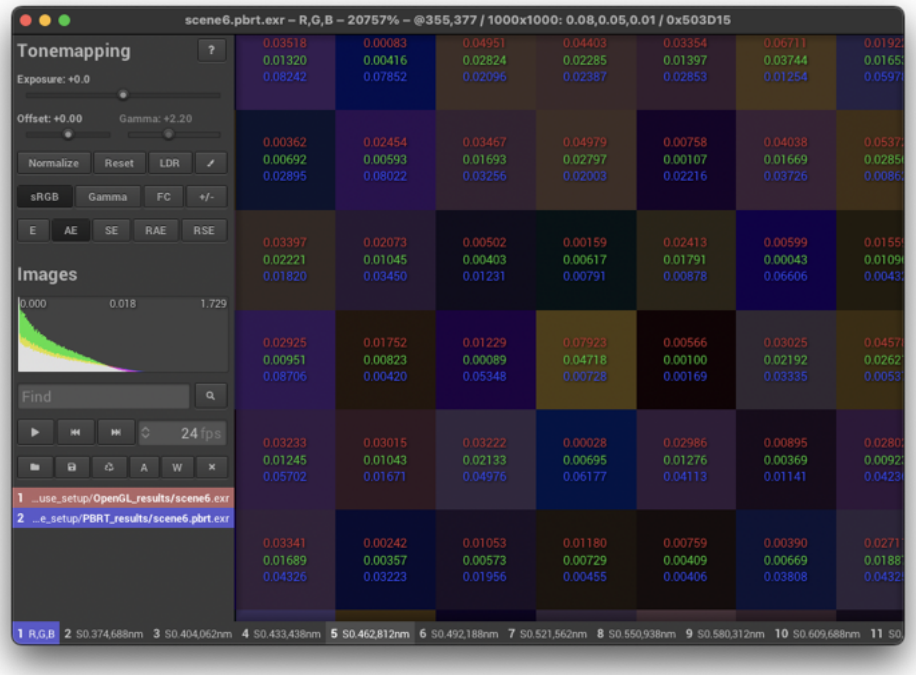


Figure 2: TEV tool interface

Furthermore, TEV can be used as a displayserver for PBRT-v4, which allows to render images directly to the TEV window. This is particularly useful for debugging purposes, as it allows to see the rendered image in real-time and make necessary adjustments to the scene or rendering settings without having to wait for the entire rendering process to be completed.

5.5 Tonemapping HDR images

High dynamic range (HDR) images are images that have a higher dynamic range than the standard 8-bit images. In case of this project, the HDR images are 32-bit floating point images. Luminance values in HDR images can be very high, and the standard 8-bit images clip the values that are higher than 255 (or 1.0 in the case of floating point values), resulting in loss of information. To display HDR images on standard monitors as well as on paper, such images need to be tonemapped.

Tonemapping is a process of mapping the high dynamic range of the HDR image to the low dynamic range of the standard image. While there are different, more complex tonemapping algorithms that can be used to compress the dynamic range of the image [15], it is not the focus to use them for this project as the goal is to display the images in a way that are informative, rather than making them exceptionally visually appealing. Therefore linear tonemapping was used for the images by simply changing the exposure of the image. *Pixelmator Pro* [16] image editing software was used for this process where exposure of an image can be modified in percentage values. To estimate the exposure adjustment, the

maximum luminance value of the image was inspected to determine the scaling factor. The scaling factor was calculated using the following formula:

$$S = \frac{1.0}{L_{max}} \tag{2}$$

Changing the exposure by a value of -100% results in the reduction of L_max by half, therefore to find the exposure value to be used to have the maximum luminance value of 1.0, the following formula can be leveraged:

$$Exposure = 100 \times \log_2(S) \tag{3}$$

After performing this dynamic range compression, images can then be saved as 8-bit images for display on standard monitors or for printing and this is the method used to display the results of this project. Images modified this way do have their exposure values stated in the image description or above the image itself.

Figure 3 shows an example of tonemapping of an HDR image. The left image is the original HDR image that is displayed with clipped values in standard dynamic range , and the right image is the tonemapped image with an exposure value of -412% where gradual luminance change is more apparent in the highlights of the image with reduced exposure.

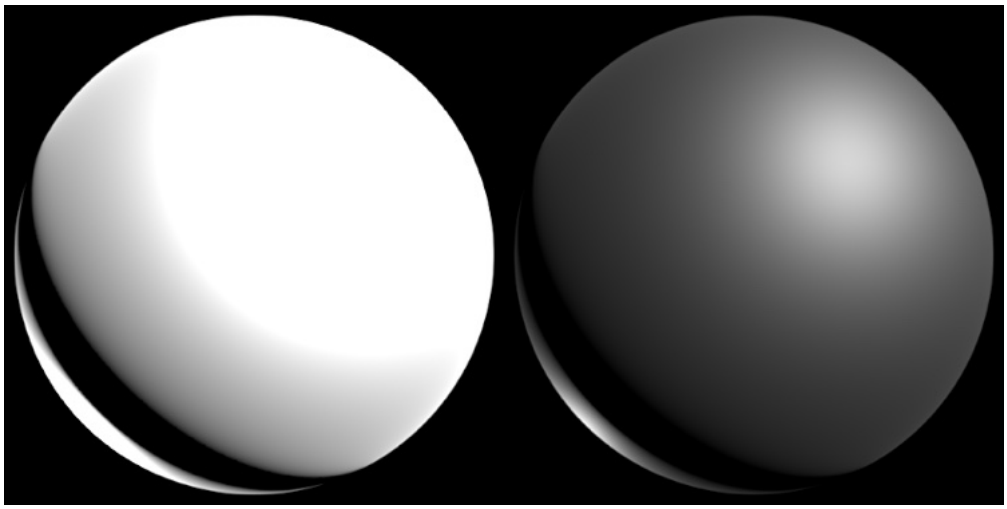


Figure 3: Tonemapping an HDR image - Original HDR image (left) and tonemapped image with an exposure value of -412% (right)

6 Diffuse Comparison

This section will describe the lighting model used for diffusion reflection, the visual comparison of the two implementations and the different sources of error that can be observed in the comparison images to provide a more accurate representation of the error between the two implementations.

Diffuse lighting model serves as an perfect starting point for analysis of differences between different rendering techniques, providing the simplest object shading scenarios acting as a groundwork for further analysis of more complex lighting models in subsequent sections of presented work and providing a reference point of expected error margin between the implementations.

6.1 Lighting model

6.1.1 PBRT-v4

PBRT-v4 simulates diffuse lighting using Lambertian reflectance model which scatters incident illumination equally in all directions. This material model *diffuse* takes a single reflectance parameter $reflectance$ that can be expressed as a spectrum or a single value in the range [0, 1] that specifies the fraction of the incidence light that is gets scattered. For the purpose of this comparison, single float values are used.

Reflection distribution function (BRDF) for Lambertian reflectance model is defined as [3]:

$$f_r(\omega_i, \omega_o) = \frac{\rho}{\pi} \tag{4}$$

where:

- ρ - reflectance parameter
- ω_i - incident light direction
- ω_o - outgoing light direction

Division by π ensures energy conservation when integrating over the hemisphere.

The contribution of the incident light to the reflected radiance is weighted by the cosine of the angle between the normal and the incident light direction, accounting for the amount of light that is incident on the surface.

Therefore the resulting radiance is calculated as:

$$L_o(\omega_o) = \frac{\rho}{\pi} \int_H L_i(\omega_i)(\omega_i \cdot n)\, d\omega_i \tag{5}$$

where H is the hemisphere above the point being shaded, L_i is the incoming radiance from the light source and n is the normal of the surface.

6.1.2 OpenGL

To implement the diffuse lighting model in OpenGL, the Lambertian reflectance model is used directly without any modifications [4]. This model is implemented in the fragment shader of the program and takes the reflectance property of the material as a parameter. The resulting radiance is calculated using the same equation as in PBRT-v4 (Equation 5).

6.2 Visual comparison

The visual comparison of the two implementations is evaluated by rendering the scene setup from Section 5 with matching material parameters, repeating the process multiple times to cover the range of possible reflectance values in 0.1 increments.

Initial visual comparison by eye shows that the two implementations are visually difficult to distinguish from each other, confirming the analogous implementation of the Lambertian reflectance model in both PBRT-v4 and OpenGL.

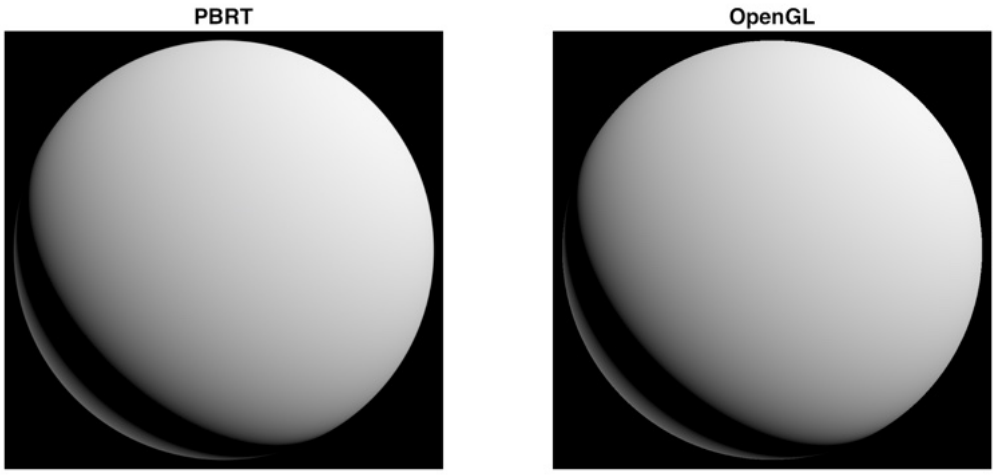


Figure 4: Visual comparison of diffuse lighting model on PBRT and OpenGL displaying an object with reflectance parameter of 0.5 . Exposure of the image is adjusted to -180%.

6.3 Quantitative comparison

Leveraging the TEV tool, the quantitative comparison of the two implementations is performed while using PBRT as the reference image. Selecting these two images allows to get the error values between the pixels as RSE and will be explored in the following subsections.

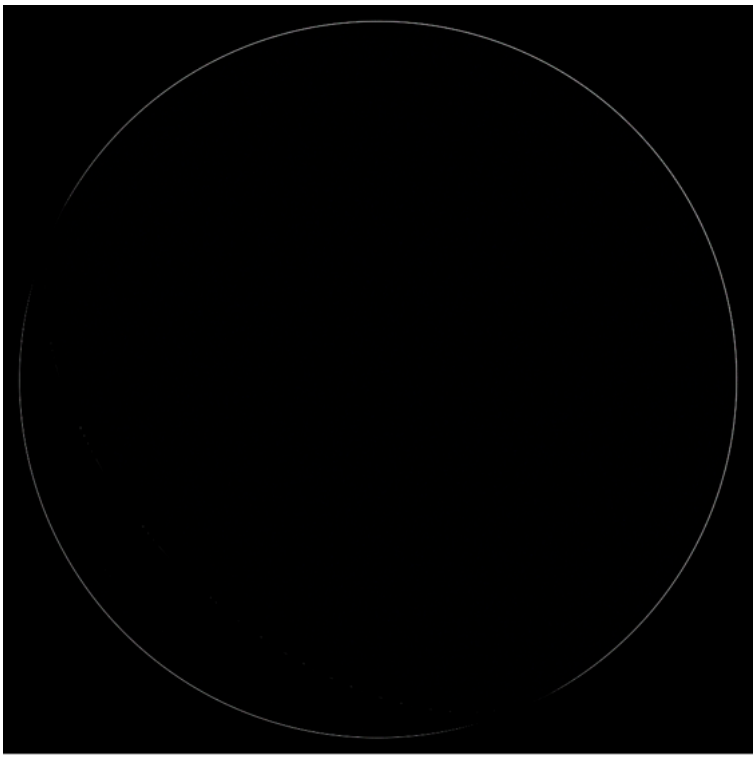


Figure 5: Difference image of diffuse lighting of PBRT and OpenGL of an object with reflectance of 0.5, using PBRT as reference image. Using -77% exposure.

6.3.1 Edge artifacts

Looking at the results of an object with reflectance parameter of 0.5, in an error metric of relative squared error, it is immediately apparent that there is a significant difference between the two images at the edges of the object as is visible in Figure 5. This mismatch is caused by aliasing (jagged edges) that are apparent to the viewer when zoomed in on the edges of the rendered object using OpenGL.

Figure 6: Zoomed in PBRT and OpenGL images of an object showing aliasing in OpenGL rendering. Exposure at -180%

The aliasing is caused by the rasterization process in OpenGL, where OpenGL calculates a single color value for each pixel on the screen. This causes the edges of the object to appear jagged and not smooth like in PBRT where the image is generated by tracing rays multiple times per pixel, providing a more accurate representation of the edges of the object.

While there are numerous solutions for mitigating aliasing in OpenGL, to verify the cause of discrepancy between the two implementations, it was chosen to make an addition of a simple anti-aliasing technique to the OpenGL implementation. The technique used is MSAA (Multisample Anti-Aliasing) which is a simple and efficient method for reducing aliasing in OpenGL. The principle behind MSAA is to finalize the color of a pixel by sampling multiple points within the pixel and averaging the results. Naturally, it should be taken into account that since multiple samples need to be evaluated for each pixel, the performance of the rendering process as well as the memory consumption will be affected and might not be suitable for all real-time applications.

Maximum number of samples used in the MSAA technique is limited by the hardware and the driver and in the case of the device used for this project, the maximum number was queried using *glGet* [17] and was found to be 4 samples per pixel which can be set to be used in the OpenGL program. The resulting OpenGL images were rendered without MSAAx4 enabled as it gives a clearer respresenation of results in the comparison images, makes the results more consistent as depending on the hardware the MSAA implementation might differ. The artifacts instead were mitigated by image filtering described in Section 6.3.4.

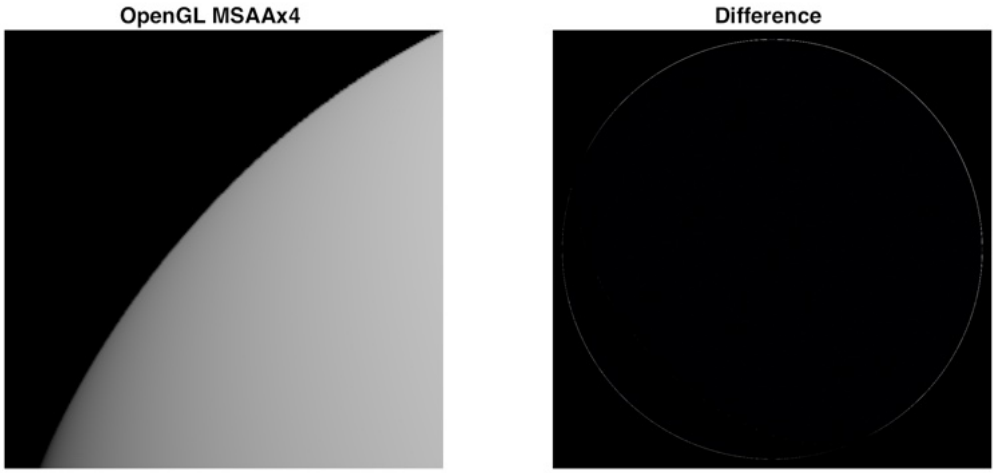


Figure 7: closeup of the rendered image with MSAAx4 enabled. Exposure at -180% (left) and comparison with PBRT (right). Exposure at -77%

While the MSAA technique does not reduce the aliasing as much as in output image of PBRT, closeup of the rendered image with MSAAx4 enabled in Figure 7 shows that the edges of the object are smoother and therefore the RSE values are are less prominent than what is observed in Figure 6. The comparison image evaluating rendered image with MSAAx4 enabled and PBRT image have been adjusted to the same exposure as the comparison image in Figure 5 to make the the reduction in error from the MSAA technique more apparent.

PBRT also includes an option to disable pixel jitter - forcing all pixel samples to be taken at the center of the pixel [11]. This essentially would make the image exibit the same aliasing as in OpenGL rendering, but was not used for this work to retain the physical accuracy of the PBRT image therefore in further comparisons in this work, pixel jitter was not disabled.

6.3.2 Sampling noise

Another source of error that can be observed in the comparison image is the sampling noise from the PBRT image. This can be made more apparent by increasing the exposure of the same comparison image, revealing this noise as seen in Figure 8.

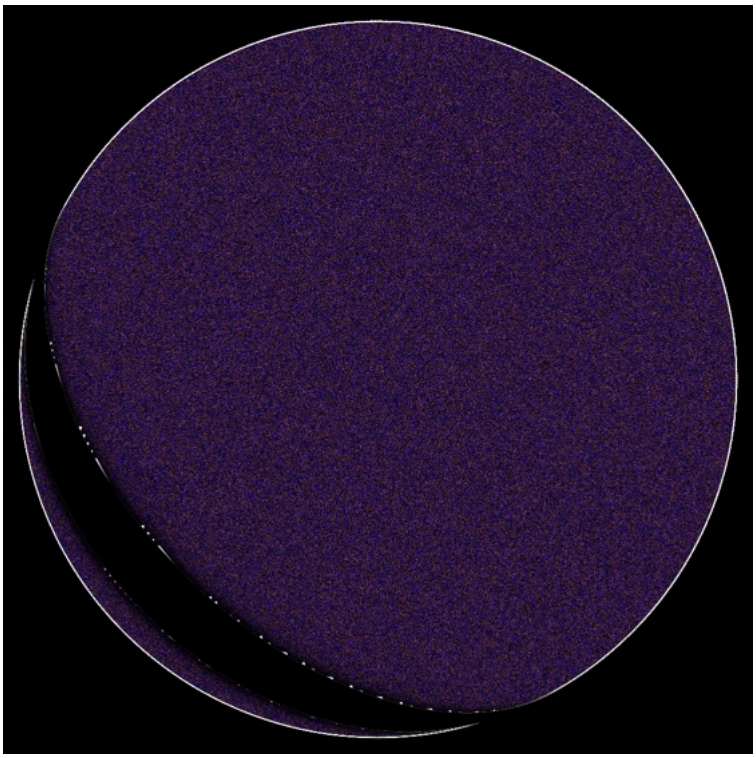


Figure 8: Difference image of diffuse lighting of PBRT and OpenGL of an object with reflectance of 0.5, using PBRT as reference image. Using 700% exposure.

This noise is a product of the Monte Carlo integration used in PBRT to evaluate the reflected radiance at a point on the object. The randomness in selecting the sample points within the pixel as well as the direction of the scattered rays, the overall sample count and sampled wavelengths per pixel influences the amount of noise in the final image. This can be mitigated by increasing the number of samples per pixel, but would severely further impact the performance of the rendering time.

Specifically the purple hue of the sampling noise in the comparison image is caused by the use of spectral rendering by PBRT to provide a more physically accurate color interactions. This is expressed as a difference of energy levels for different color wavelengths in the spectrum. By default PBRT uses spectral response curve of CIE 1931 illustrated in Figure 9. From the response curve it can be seen that the purple hue is caused by the higher energy levels of the blue and red wavelengths in the spectrum.

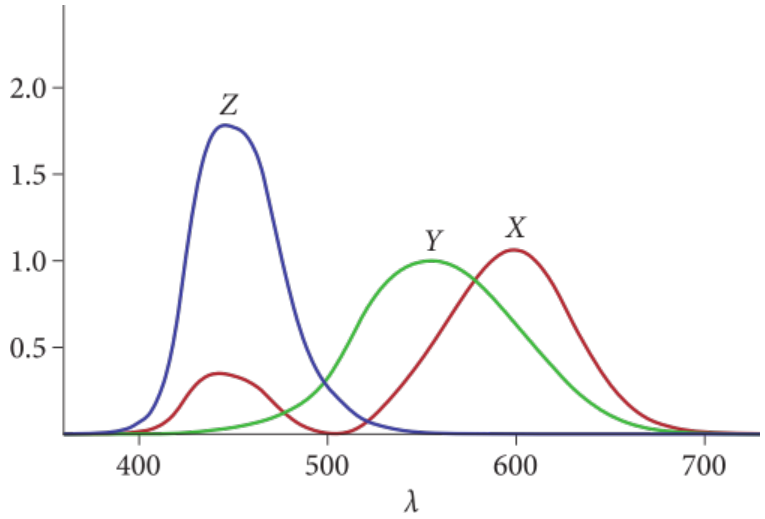


Figure 9: CIE 1931 color matching curves. [3, Section 4.6.4]

PBRT has an option of disabling wavelength jitter - this forces all the samples for a pixel to be sampled at the same wavelengths [11]. This would effectively remove the noise, but also introduce an unpleasant color error to the image affecting the results even further and therefore will not be used in this work.

6.3.3 Self-Intersection artifacts

Looking closer at the comparison image in Figure 10, it can be observed that at the points where the incoming light direction and the surface normal are nearly perpendicular, there are visible artifacts in the PBRT image. These artifacts are particularly prevalent in such areas and are caused by the ray intersecting the surface that it is being traced from, causing the surface to shadow itself. This is a common issue in ray tracing due to numerical precision issues [3] and could be mitigated by using a number of techniques such as adjusting the epsilon value (a small offset added to the ray origin to prevent self-intersection) in the ray tracing implementation.

Unfortunately, this epsilon value is not exposed for the possibility of adjusting it in PBRT scene description files, and requires recompilation of entire PBRT source code to adjust it. Its ideal value depends on the scale of the scene and therefore it is not possible to provide a universal value that would work for all scenes. Additionally, since PBRT is not the renderer of interest to perfect the results in, it was chosen to not further investigate this issue and instead to filter out these artifacts from the comparison image as described in 6.3.4.

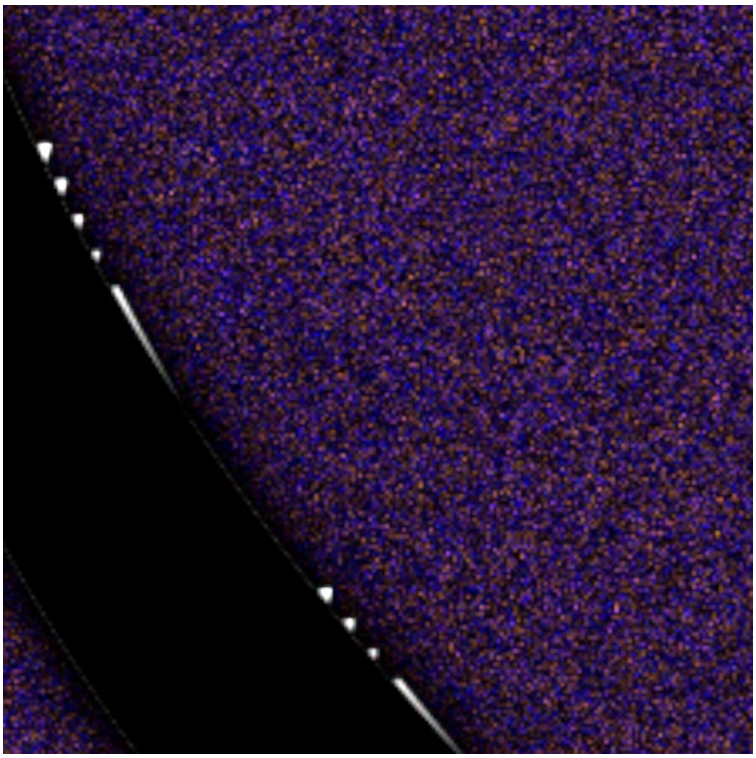


Figure 10: Zoomed in difference image of diffuse lighting of PBRT and OpenGL of an object with reflectance of 0.5, using PBRT as reference image. Using 700% exposure.

6.3.4 Result filtering

To provide a more meaningful representation of the error between the images rendered by PBRT and OpenGL, the images showing the RSE error for the difference in the images were filtered to remove the background, edge artifacts and self-intersection artifacts.

Since the background of the images is black, it skews the error values in the comparison image, providing a false representation of the error. Therefore the background of the comparison image had its alpha channel reduced to 0, effectively removing the background from the image. This was done by applying a mask equivalent to the radius of the sphere around the center of the image.

Then, to tackle the edge artifacts and the self-intersection artifacts, it was chosen to filter them out from the comparison image. To filter the unneeded values, initially, the treshhold was found by calculating the mean value and the standard deviation of the image, and then using the mean value plus the standard deviation as the threshold value. This did not provide the desired results, as insufficient amount of the artifacts were removed from the image. Therefore, it was chosen to use otsu's thresholding method (binary thresholding) to find the threshhold value that would more closely separate the image into two parts - artifacts (which have a larger magnitude) from the rest of the image (which has a smaller magnitude). To make the thresholding more accurate, median filter (of a size of 4x4) was applied to a copy of the image smoothing out the noise, making the pixel values along the edges more uniform. After that, 8bit grayscale conversion was applied to the image and

the threshold value was found using otsu's method. It works by trying all possible (8bit) threshold values to divide the image and selecting the one that minimizes the intra-class variance (how similar the pixel values are in the same class) [18]. This found threshhold value is then scaled back to the original 32bit float value and used to create a mask for the original image where the artifacts would effectively be removed from the image.

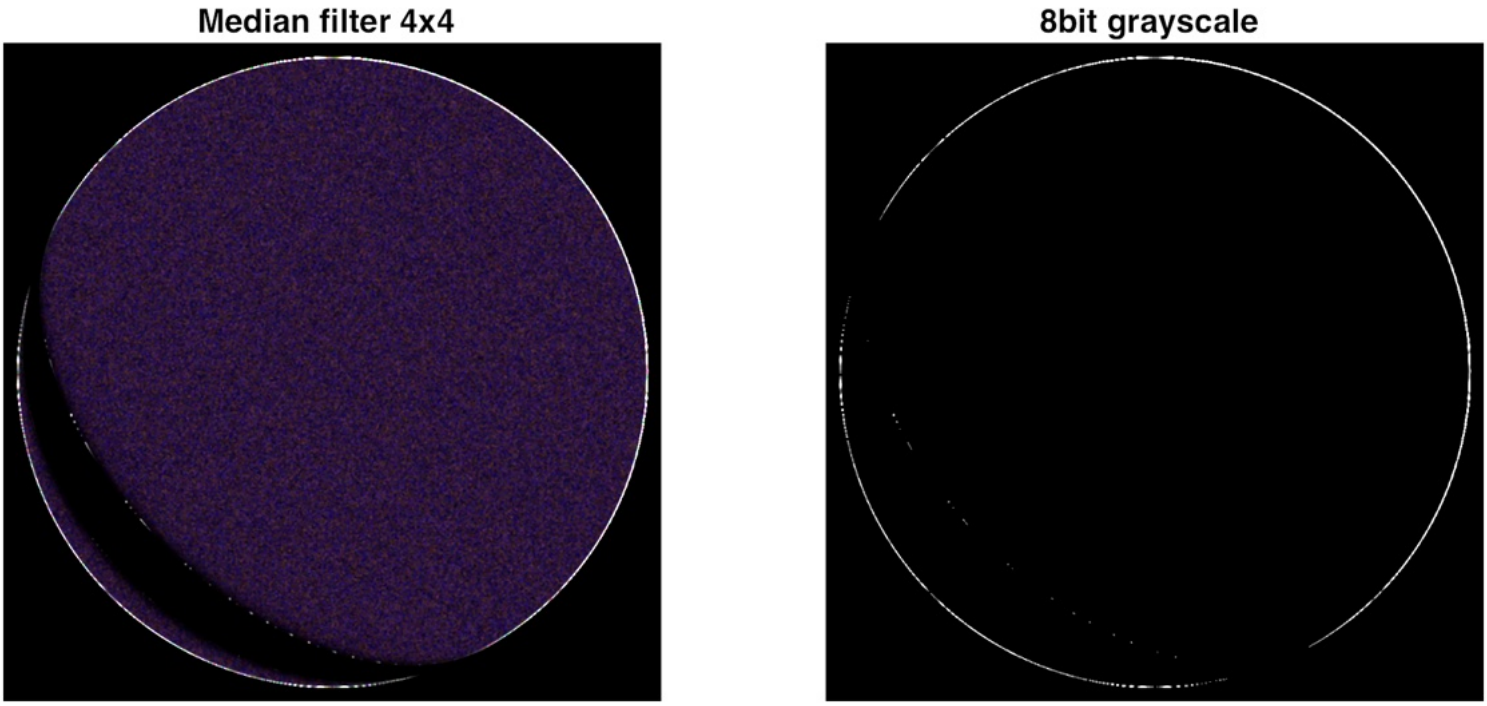


Figure 11: Difference image of diffuse lighting, with a median filter applied (left) and converted to an 8bit grayscale image (right) Using 700% and 900% exposure respectively.

Comparing the original image with the filtered image, it can be seen that the largest artifact values skewing the overall RSE of the image have been removed, retaining only the values that are very close to the existing sampling error and not removing a lot of the actual error between the images. Additionally, the image is also cut to the size of the sphere, to reduce the amount of black pixels, skewing the error values.

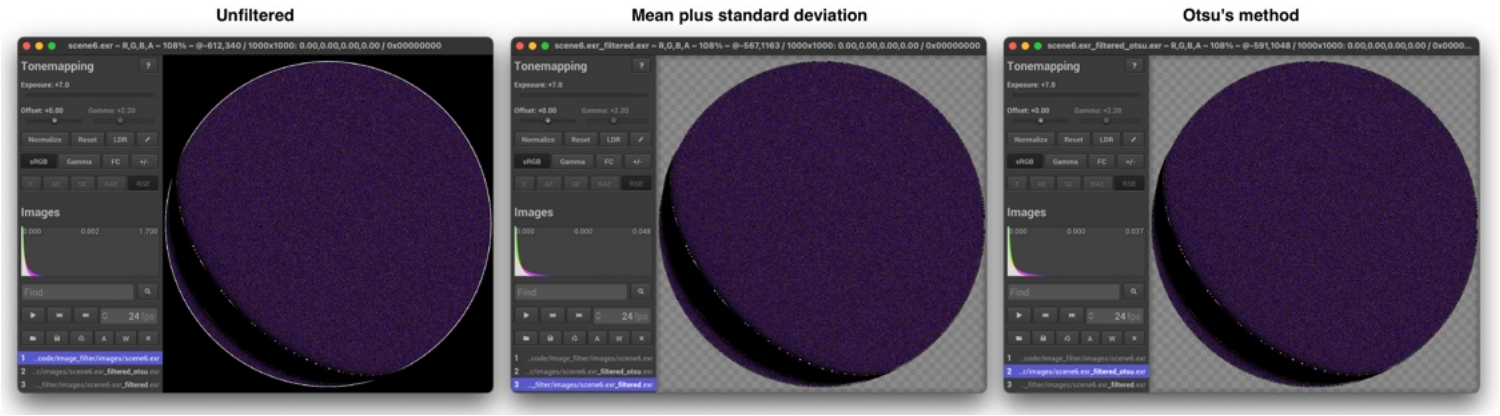


Figure 12: Difference image of diffuse lighting, with the artifacts filtered out compared to original unfiltered difference image. Using 700% exposure.

Unfortunately, while the filtering process was an improvement over using mean and standard deviation thresholding, the results did not turn out to be consistent across images with different reflectance values as the maximum error values after filtering would be significantly different for each image. Changing the size of the median filter for different images does make the result more in line with the expected filtering results, but requires manual adjustments for each image. Therefore, Otsu's threshold method was not used to clean up the images for further analysis.

Downfall of this method is likely precision issues having an impact finding a threshold value as the image has to be converted to 8bit grayscale image, which depending on the image can cause a big loss of information when there are large differences between the pixel values, which is the case for a lot of images being of RSE values.

Best results were achieved by slightly reducing the radius (by 2 pixels) of the mask used to remove the background of the image which then removes the aliasing artifacts from the edges of the object, while keeping the imporant pixel values for comparison intact. Applying mean and standard deviation threshold to the image with the modified mask provided adequate filtering across different reflectance values with consistent maximum RSE values as well as when used to other models comparisons in the following sections. Images portraying larger error values were not filtered and instead just cropped to the size close to the object to remove the background and the edge artifacts.

6.4 Error analysis

The error analysis was performed by evaluating the RSE values of the images with different reflectance values. The Full visual comparison of the images with different reflectance values can be seen in Figure 14. All rendering results have their exposure adjusted to accomodate the image with the highest luminance values to not have the image values clipped. The difference images have their exposure adjusted to a high value to make the error values more visible.

Extracted mean and maximum RSE values from the filtered images were plotted against

the reflectance values of the objects to see if and how the error changes with the reflectance value, establishing the minimum error possible error that can be achieved between the two renderers as the Lambertian reflectance model is used equivalent in both implementations and ideally should produce the exact same results. The error graph can be seen in Figure 13.

By extracting mean and maximum RSE values from the filtered images, the error values were plotted against the reflectance values of the object to see if and how the error changes with the reflectance value, thereby establishing the minimum achievable error. Both implementations utilized the Lambertian reflectance model, which theoretically should yield identical results. Figure 13 illustrates this error graph.

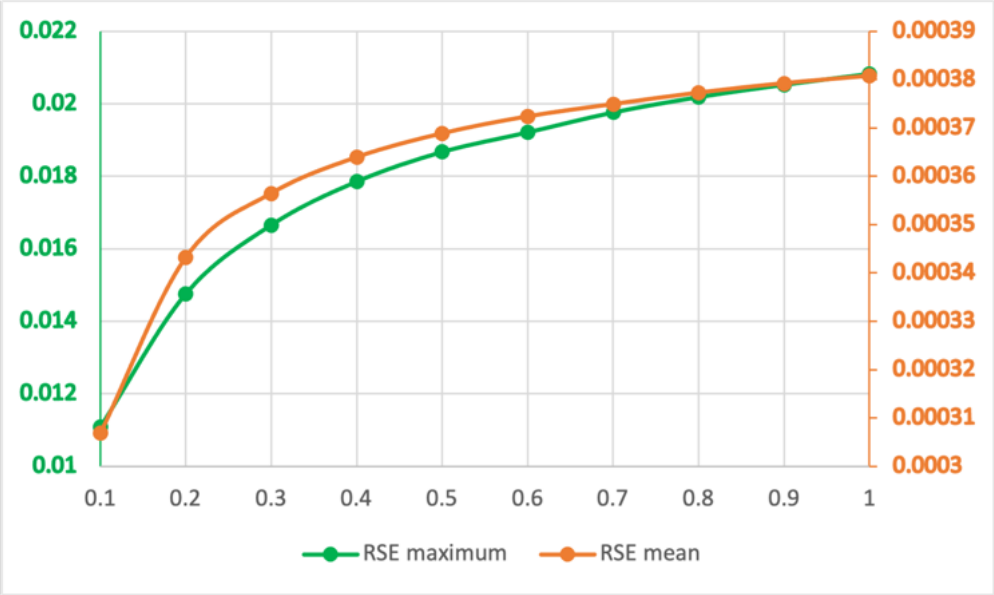


Figure 13: Graph showing the mean and maximum RSE values of the images with different reflectance values from 0.0 to 1.0 in 0.1 increments.

Due to the nature of the Lambertian reflectance model, the error values should be consistent across different reflectance values, as the model scatters the incident light equally in all directions, regardless of the reflectance value. The graph does show a non linear increase in error values across different reflectance values, error likely due to 0.01 added to the reference value for RSE calculation to avoid division by zero.

Uneven filtering results are likely to also contribute to the non-linearity of the error values across different reflectance values.

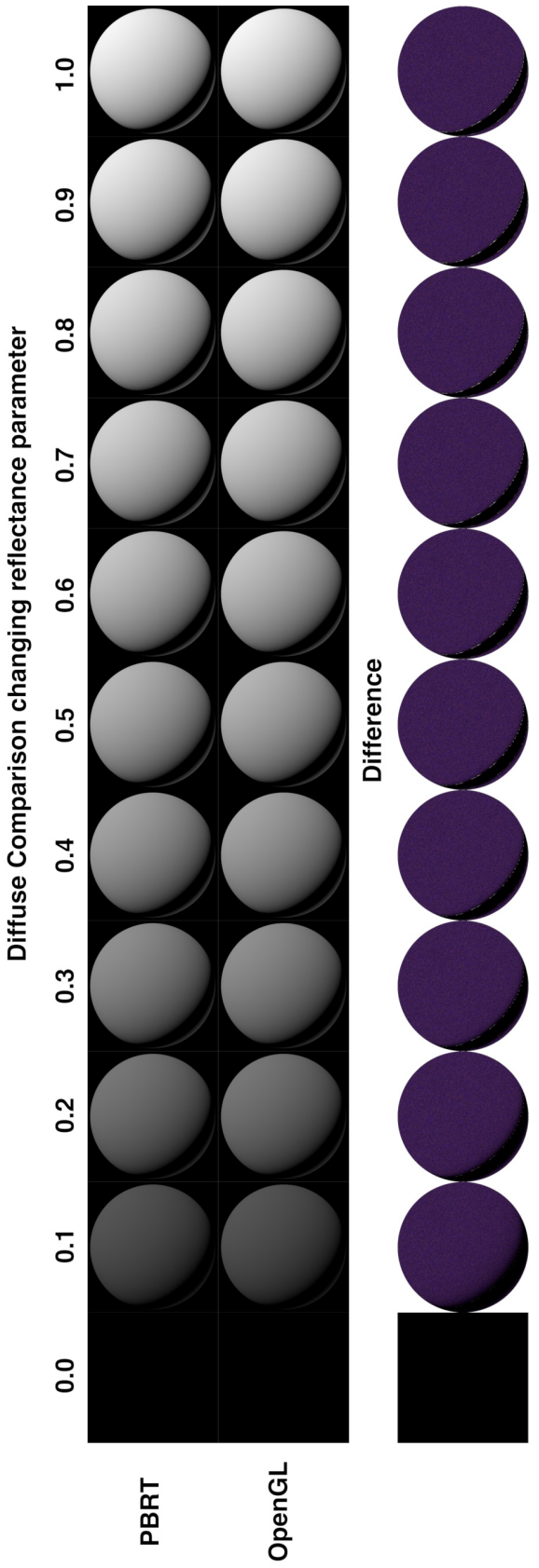


Figure 14: Rendered images of the object with different reflectance values in PBRT and OpenGL, and the difference images showing the RSE values between them. Rendered images have exposure adjusted to -280% and difference images to 700%.

7 Specular Comparison

This section explores physically accurate representation for specular reflections and evaluates the correctness of the specular reflection implementation in OpenGL by comparing it with the results from PBRT.

Most real objects are not perfectly diffuse, but have some specular (mirror-like) reflection. There are multiple models that can be used for the specular reflection, such as the Phong model where specular component is spread out around the specular direction using a cosine function raised a power or the Blinn specular model which accounts for situations where view and reflection directions are greater than 90 degrees apart. These reflection models are widely used in real-time applications due their simplicity and computational efficiency. These models consist of three components: ambient, diffuse and specular. The ambient component is the light that is uniformly incident from the environment, the diffuse component is the light that is scattered uniformly in all directions and the specular component represents the highlights - light that is reflected in a mirror-like way [19]. While these models can create visually pleasing and realistic results for certain types of materials such as plastics or some metals, these methods, however, are not representative of real-world materials and therefore such results do not align with the refference renderer - PBRT as different parameters are used to evaluate the specular reflection.

7.1 Lighting Model

To create a more physically accurate specular reflection, the simple specular model was replaced with Cook-Torrance model. Using this model, the intensity of the reflected light is evaluated from the intensity of the light source and reflectivity and roughness of the material. For this model, the specular component is assumed to be a surface consisting of small individual faces - microfacets, meaning tat normals of each microfacet might not be the same as the normal of the surface, depending on the roughness of the material. These microfacets are then described by two statistical measures, a microfacet distribution function D and a shadowing-masking function G. Specular component of this model is calculated using the following formula [19]:

$$R_s = \frac{F}{4} * \frac{D * G}{(N \cdot L)(N \cdot V)} \tag{6}$$

Where:

- F - Fresnel term
- D - Microfacet distribution function
- G - Shadowing-masking function
- N - Surface normal vector
- L - Light vector

- V - View vector

The Fresnel term for this comparison was kept at 1.0 and will not be taken into account for this comparison, but is further incorporated in the model described in section 8.

Microfacet distribution function describes the statistical distribution of surface normals over a microsurface. There are multiple distribution functions that can be used for this calculation, such as Beckmann, Phong or GGX that can be utilized based on the desired accuracy and performance [20]. For this comparison, it was chosen to use Trowbridge-Reitz GGX normal distribution function which matches the function used in PBRT calculations. The GGX distribution function is defined as follows [20]:

$$D = \frac{\alpha^2}{\pi((N \cdot H)^2(\alpha^2 - 1) + 1)^2} \tag{7}$$

Where:

- α - Roughness parameter
- N - Normal vector
- H - Half vector

Half vector is calculated as follows:

$$H = \frac{L + V}{||L + V||} \tag{8}$$

Shadowing-masking function describes the fraction of microfacets that are visible from both the direction of the light and the direction of the viewer. Its impact is most prevalent in the case of rough surfaces and near grazing angles. There are multiple shadowing-masking functions that can be used, such as Beckmann, GGX, Kelemen or Smith. For this comparison, it was chosen to use the Smith shadowing-masking function that matches the function used in PBRT. The Smith shadowing-masking function is defined as follows [3]:

$$G = \frac{1}{1 + \lambda(\omega_o) + \lambda(\omega_i)} \tag{9}$$

and λ is an auxilary function measuring invisible masked microfacet area per visible microfacet are and is defined as:

$$\lambda(\omega) = \frac{(-1 + \sqrt{1 + \alpha^2 * tan^2(\theta)})}{2} \tag{10}$$

Where:

- α - Roughness parameter
- θ - Angle between the normal and the vector

The roughness parameter for these calculations is a scalar value that describes the roughness of the material. This value is of range [0.0,1.0], where 0.0 represents a perfectly smooth surface and 1.0 represents a perfectly rough surface. The roughness parameter, is used in both the GGX distribution function and the Smith shadowing-masking function influencing the intensity of perceived specular reflection.

7.1.1 PBRT

In PBRT-v4, a material containing was created by setting a type *conductor* with a reflectance value of 1.0 based on which plausible values of *eta* and k are then calculated, resulting in a fully reflective material containing no diffuse component. In addition, the material takes the parameter specifying surface roughness. To verify the utilization of Cook-Torrance reflectance model in PBRT, the rendering system was built using debug build [21]. This build allows to set breakpoints in the code, follow the execution of the program and observe the values of the variables at each step. Calculations were then verified by setting breakpoints in the code and following the execution of *TrowbridgeReitzDistribution* class functions in *scattering.h* file [22], containing the microfacet distribution function and the shadowing-masking function. The results were then compared to the calculated values using OpenGL implementation.

7.1.2 OpenGL

In OpenGL, specular reflection can be calculated using the same Cook-Torrance model as in PBRT. The resulting radiance is calculated as follows:

$$L_o = \frac{F}{4}\frac{D * G}{(N \cdot L)(N \cdot V)} L_i (N \cdot L) \tag{11}$$

Where:

- L_o - Outgoing radiance
- L_i - Incoming radiance

7.2 Visual Comparison

The visual comparison of the specular reflection is evaluated using the same scene as in the previous comparison with renders having matching material parameters, in this case - material roughness. Different roughness values were used to evaluate the correctness of the specular reflection implementation. Visually comparing the results, no immediately apparent differences can be observed by a naked eye, implying successful implementation of the Cook-Torrance model in OpenGL.

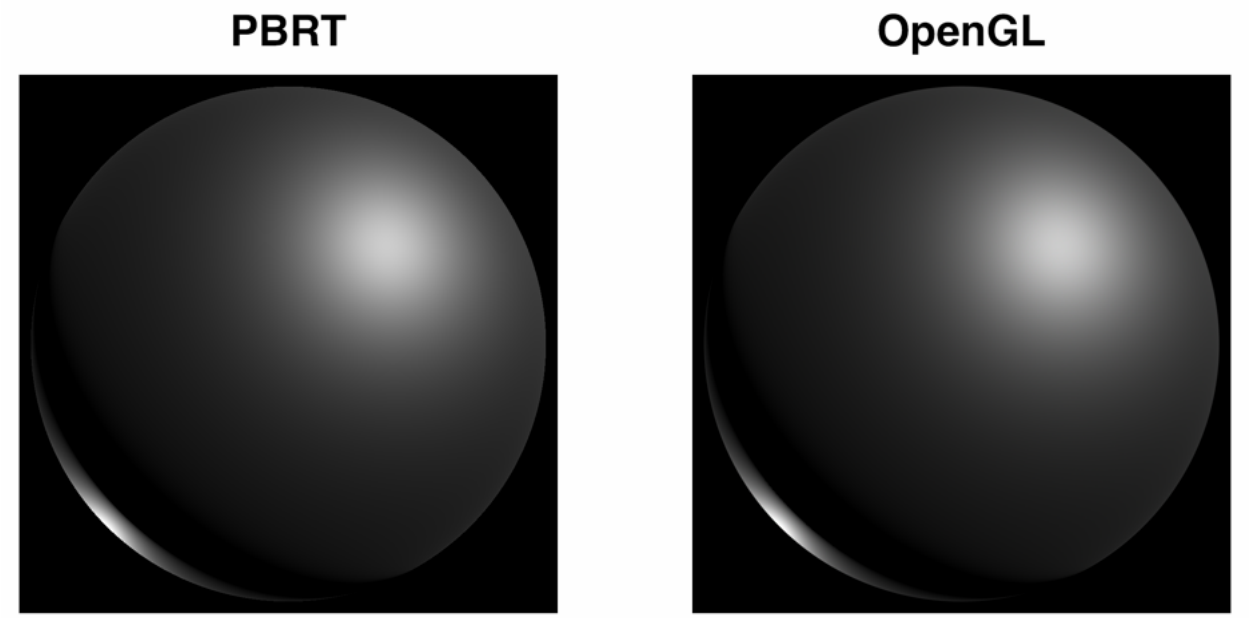


Figure 15: Rendered images of the object with roughness value of 0.3 in PBRT and OpenGL. Exposure of the images is adjusted to -510%.

7.3 Quantitative Comparison

Looking at the image comparison results from TEV, results are compared using PBRT rendered image as a refference and RSE metric is used to evaluate the difference between the images.

Using the 32bit float values for the color buffer in both render images, was specifically useful for this model as the color values in the highlights when using low roughness values can be extremely high, and using 8bit color buffer would result in a loss of precision and information. Besides that, this high intensity of the highlights is more sensitive to errors in the calculations, and proved to be useful to further refine small errors of implementation, such as exposing lack of normalization of interpolated normal vectors in the fragment shader causing visible artifacts in the image. This can be seen in the image 16.

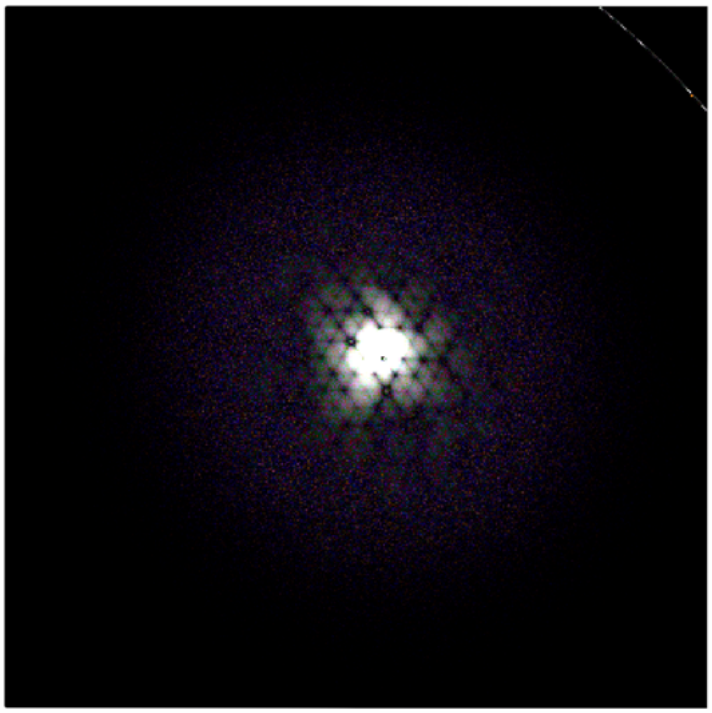


Figure 16: Comparison image showing the artifacts stemming from OpenGL image at the highlight of the object due to lack of normalization of the interpolated normal vectors in the fragment shader. Roughness of the object is 0.01, exposure of the image is adjusted to 500%.

7.4 Error analysis

The image showing RSE between the pixel values of two images was filtered using method explained in 6.3.4 to remove unwanted artifacts affecting the results.

Images showing the RSE values of the images with different material roughness values can be seen in Figure 21. The images have their exposure adjusted to a high value to make the error values more pronounced. Looking at the images, the error values are concentrated around the area of the highlights, with the area spreading out and intensity of the error decreasing as the roughness value is adjusted to a higher value. The observed error is the sampling error that is present in the PBRT image, much like in the previous comparison, which is more pronounced for the specular model as the intensity of the highlight is higher and is therefore more complex to sample adequately. This is somewhat alleviated by PBRT already incorporating multiple importance sampling to weigh the samples using probability density functions [3], but as the highlight is very small and intense, it is still exhibits noticeably higher error values. This can further be attempted to alleviate by increasing the number of samples used in the rendering process, but this would increase the rendering time significantly.

Mean and maximum RSE values from the comparison images were plotted against the

roughness values of the material, to observe the change in error values as the roughness of the material is adjusted. The results can be seen in 17.

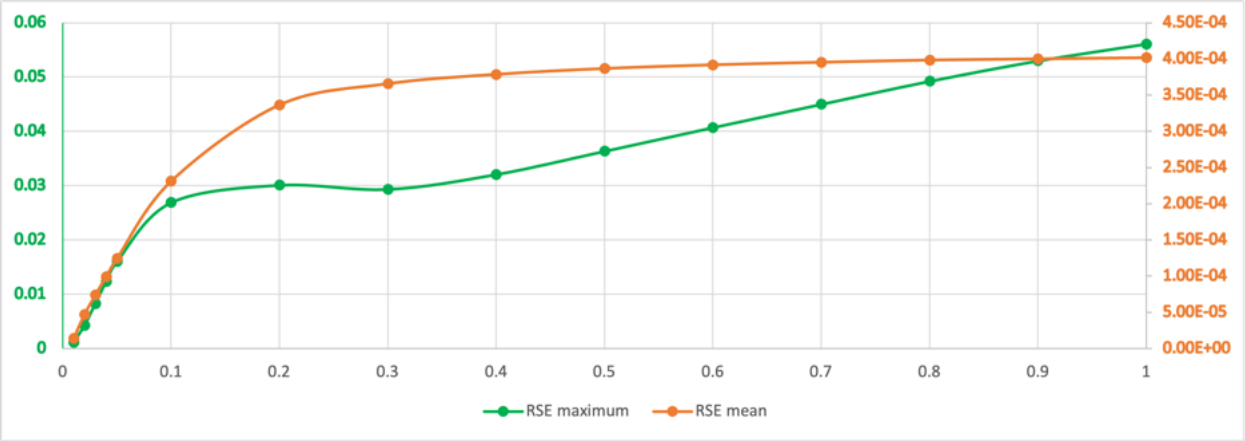


Figure 17: Graph showing the mean and maximum RSE values of the images with different roughness values from 0.01 to 1.0 with increments of 0.01 (from 0.01 to 0.05) and 0.1 (from 0.1 to 1.0).

The graph shows that the mean and maximum RSE values are close to zero when roughness is a very small (0.01), with the values increasing as the roughness is increased. Considering that minimal roughness has a small, but very bright highlight, the maximum RSE value being a small value was not expected, therefore the results were further analyzed.

The cause of this inconsistency was due to RSE around the area of that small highlight being so high - especially at the point where the highlight is from the grazing angle (bottom left corner), that it was over the the value used to filter the image 6.3.4. This resulted in RSE values at the center of the highlight being being filtered out, impacting the observed results. This is shown in the figure 18 (middle image).

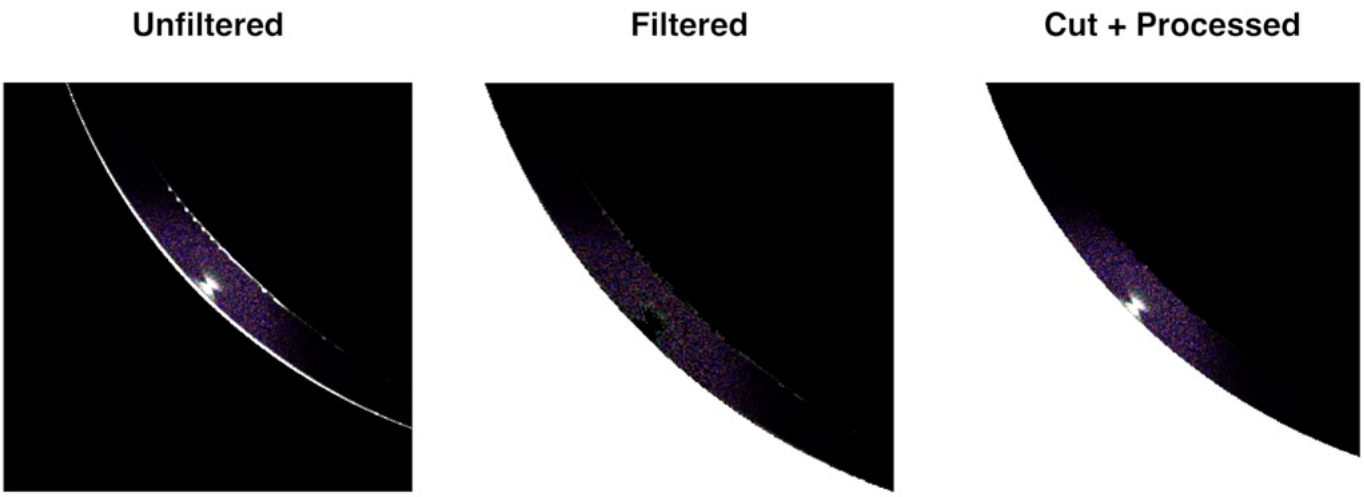


Figure 18: Three images showing section of the image with the highlight from the grazing angle using roughness value of 0.01 and 0.1. The first image shows the RSE values of the image before filtering, the second image shows the RSE values of the image after filtering and the third image shows the processing of the image manually.

To avoid using unfiltered image to represent mean and maximum RSE values as it would represent skewed results due to previously described artifacts in section 6.3, the image was manually processed to remove unwanted artifacts. The image was circularly cut to an even smaller radius to remove edge artifacts and the self-intersection artifacts were removed on a pixel by pixel basis. An example of the manually processed can be seen in the third image of figure 18.

Error graph was then plotted using the manually processed image, the results of which can be seen in figure 19. Loooking at the results, the maximum RSE values now correctly correspond to the the values observed in the image itself, with the mean RSE values not being affected by the change too much due to small area of the highlight and other pixels values within the circle being close to zero.

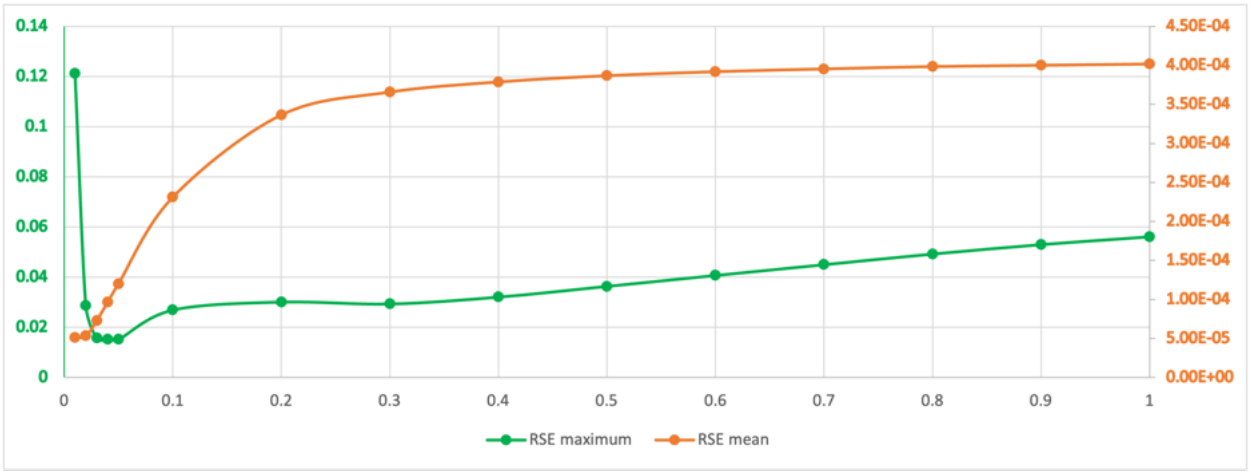


Figure 19: Corrected graph from Figure 17 showing the mean and maximum RSE values incldluing manually corrected images for roughness values from 0.01 to 0.05.

This correction correctly visualises the error amount perceivable for very small roughness values. The error for the highlight at a grazing angle (bottom left) is significantly higher when compared to the highlight slightly to the top right of the sphere, as at this more direct angle the highlight has lower variance and is more consistent, resulting in lower error values. Higher sampling values for PBRT rendering would be required to reduce this error to a minimum, but the higher roughness values adequately prove the correctness of the implementation.

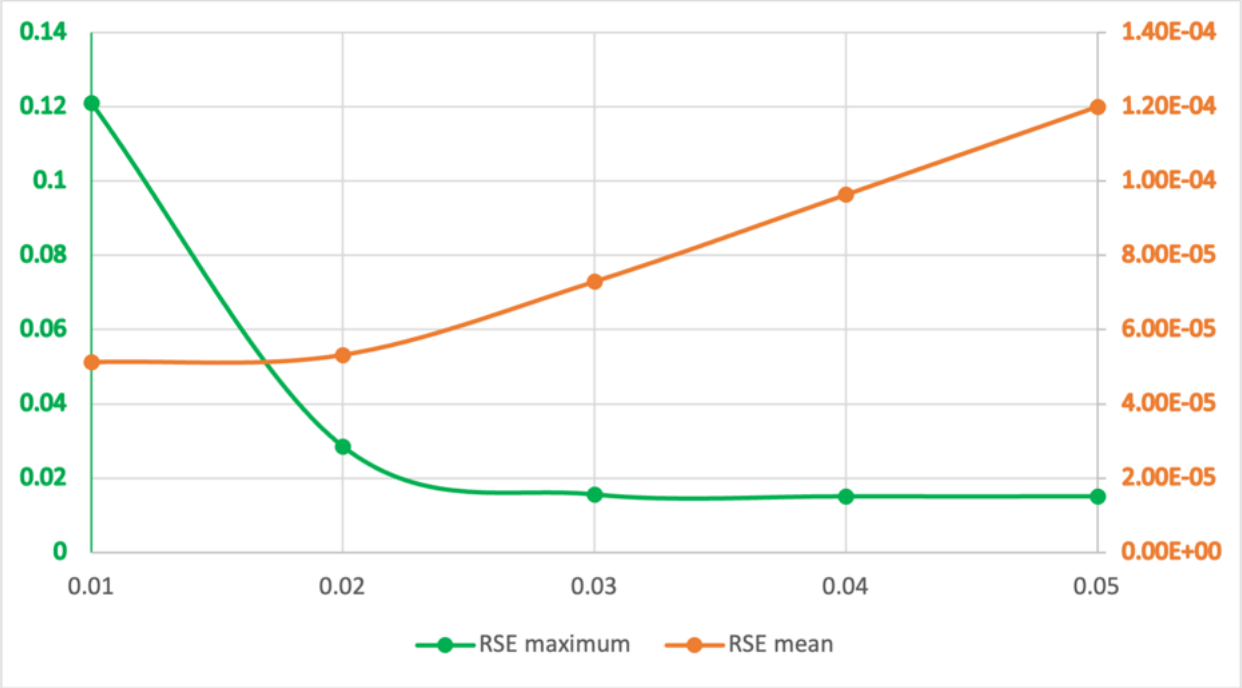


Figure 20: Corrected graph from Figure 19 showing the mean and maximum RSE values for roughness values from 0.01 to 0.05.

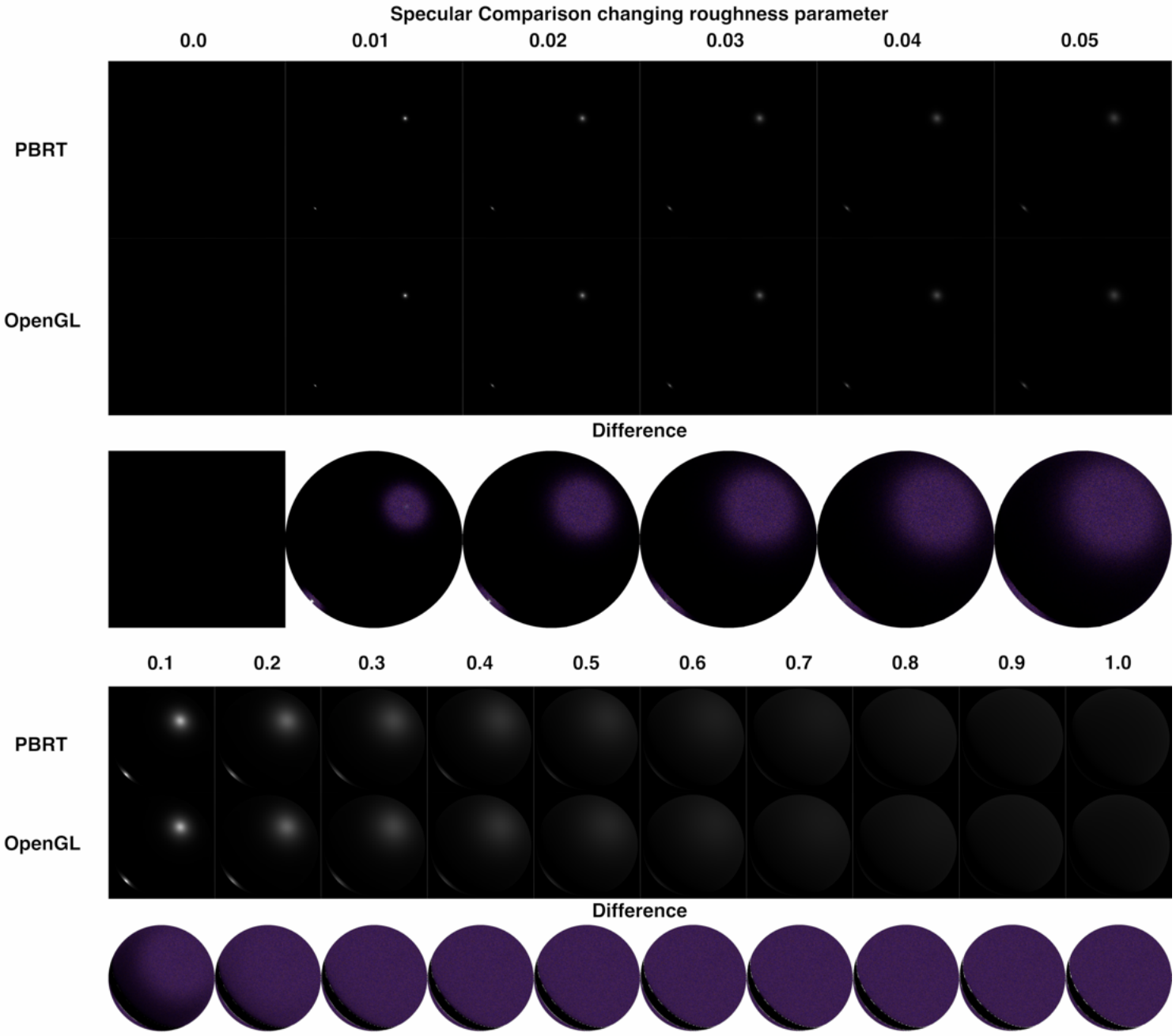


Figure 21: Rendered images of the object with different roughness values in PBRT and OpenGL, and the differnce images showing the RSE values between them. Rendered images for roughness values of 0.01 to 0.05 have exposure adjusted to -1400% and for roughness values of 0.1 to 1.0 to -860%. Comparison images have their exposure adjusted to 700%.

8 Subsurface Comparison

This section will first cover the theory behind subsurface scattering - the lighting model used to represent it, it's implementation in PBRT and OpenGL, and the visual and quantitative comparison of the two methods followed by the analysis of the results. As there are three iterations of implementation of subsurface scattering in OpenGL, distinct subsections will be dedicated to compare each of them to the PBRT implementation.

Many shading models focus solely on how light interacts with an object's surface. However, in reality, many materials are not entirely opaque, allowing light to penetrate and scatter beneath the surface. This phenomenon is known as subsurface scattering. Subsurface scattering describes the process where light penetrates an object's surface, scatters within it, and then either gets absorbed or exits the object. This effect significantly influences the appearance of materials like skin, marble, milk, wax, and many others [5] [23].

For granular materials, subsurface scattering is particularly important because individual grains are small, allowing light to penetrate and scatter within the object. This scattering greatly affects the appearance of materials with distinct crystal-like grains, such as sugar, salt, sand, and snow. Understanding and implementing subsurface scattering in granular materials is crucial for achieving more realistic light transport as in such environments light penetrates the surface layer of the grain, scatters deeper within, resulting in a smoother, more cohesive appearance of the aggregate [24].

8.1 Lighting Model

This section will discuss the lighting model used to represent subsurface scattering. It will cover processes affecting the distribution of radiance within a medium, the Radiative Transfer Equation (RTE) necessary to simulate light transport within a medium, and how it is incorporated into the Rendering Equation to account for light interactions at the surface and within the medium. Additionally, it will explain the bidirectional surface scattering distribution function (BSSRDF) model used to generalize reflection and transmission properties of a surface and the components of the BSSRDF model - diffusion approximation and single scattering.

Previously, in rendering a scene, we assumed that light interacts solely with the surface of an object. However, to accurately depict subsurface scattering, it is essential to simulate light passing through a participating medium, in this case, the object itself to account for light interactions with particles within the object. This approach is also what enables the rendering of atmospheric effects such as haze, beams of light in fog, and clouds.

There are three main physical processes affecting distribution of radiance in an environment with participating media:

- Absorption - The reduction in radiance due to light converting to other form of energy e.g. heat. Absorption coefficient σ_a quantifies the absorbed per unit distance.
- Scattering - The redirection of light due to interaction with particles in the medium. the amount per unit is measured by the scattering coefficient σ_s.

- Emission - Light generated by the media itself.

For our purposes of representing granular materials, the emission process can be ignored as the grains are not expected to emit light.

8.1.1 Absorption

Absorption is a fundamental interaction of light with media such as smoke. It occurs when particles in the medium absorb the light and convert it to other forms of energy, such as heat. The absorption coefficient σ_a quantifies the rate of absorption per unit length of travel through the medium.

Mathematically, the change in radiance along a ray over a differential length can be described by the following differential equation [3]:

$$dL_o(x,\omega) = -\sigma_a(x,\omega)L_i(x,-\omega)dt \tag{12}$$

8.1.2 Out scattering

Out scattering is the process of light scattering out of the object as ray passes through a medium. This phenomenon reduces the intensity of the light as it passes through the medium. It is described by the scattering coefficient σ_s which quantifies the rate of scattering per unit length of travel through the medium. Similar to absorption, the change in radiance along a ray length dt can be described with the following differential equation[3]:

$$dL_o(x,\omega) = -\sigma_s(x,\omega)L_i(x,-\omega)dt \tag{13}$$

The total reduction of radiance is described by the extinction (attenuation) coefficient σ_t which is the sum of the absorption and scattering coefficients [3]:

$$\sigma_t = \sigma_a + \sigma_s \tag{14}$$

Additionally these coefficients define single scattering albedo ρ which is the ratio of scattering to extinction [3]:

$$\rho = \frac{\sigma_s}{\sigma_t} \tag{15}$$

8.1.3 In scattering

Opposite to out scattering, in scattering accounts for increase in radiance as light scatters into the object. Mathematically, the change in radiance along a ray length dt can be described with the following differential equation [3]:

$$dL_o(x,\omega) = \sigma_t(x,\omega)L_s(x,\omega)dt \tag{16}$$

where $L_s(x,\omega)$ is the radiance scattered into the object and calculated as follows:

$$L_s(x,\omega) = \frac{\sigma_a(x,\omega)}{\sigma_t(x,\omega)} L_e(x,\omega) + \frac{\sigma_s(x,\omega)}{\sigma_t(x,\omega)} \int_{S^2} p(x,\omega_i,\omega) L_i(x,\omega_i) d\vec{\omega}_i \tag{17}$$

In this equation:

- $L_e(x,\omega)$ is the radiance emitted by the object.
- $L_i(x,\omega_i)$ is the radiance incident on the object.
- $p(x,\omega_i,\omega)$ is the phase function which describes the distribution of light scattered from the object.

For this project, the emission term can be ignored as the grains are not expected to emit light.

8.1.4 Transmittance

Having defined the processes of absorption, out scattering and in scattering, it is then essential to define the transmittance function. In Monte Carlo path tracing such as that used in PBRT, the transmittance function is used to account for the scattering and absorption of light as it passes through the medium. For a ray passing through a medium, the transmittance function is defined as [3]:

$$T_r(p,p') = e^{\left(-\int_p^{p'} \sigma_t(p+t\omega),dt\right)} \tag{18}$$

where p and p' are the points along the ray, ω is the direction of the ray and σ_t is the extinction coefficient.

This function enables the path tracer to simulate not only the surface interactions of light but also the complex internal interactions within the medium. By accounting for these internal interactions, the transmittance function facilitates the realistic rendering of effects such as subsurface scattering, where light penetrates, scatters, and either gets absorbed or exits the material.

8.1.5 Radiative Transfer Equation

The overall equation that describes the behavior of light as it interacts with a medium is the radiative transfer equation (RTE), accounting for the absorption, scattering, and emission of light within the medium. This is the equation a path tracer solves to simulate the light transport within the medium. The equation is as follows [3]:

$$L_i(p,\omega) = T_r(p_s \to p) L_o(p_s,-\omega) + \int_0^t T_r(p' \to p) \sigma_t(p',\omega) L_s(p',-\omega)\, dt' \tag{19}$$

where p' is a point along the ray and p_s is the point where the ray intersects the surface of the object. First term on the right side of the equation accounts for reflected radiance from the surface of the object the ray intersects, while the second term accounts for the addition

of radiance along the ray due to volumetric scattering and emission up to the intersection point on a surface of the object. from the perspective of rendering granular materials, RTE is used to simulate the light transport within the grains, as well as to account for light passing through multiple grains into deeper levels to account for more realistic indirect lighting effects.

while RTE effectively models transport of light within a medium, it does not, however, handle the interaction of light at the surface of the objects such as reflection, refraction or subsurface scattering. These interactions are expanded with the rendering equation and a Bidirectional Scattering Distribution Function (BSDF) generalizing reflection and transmission properties of a surface.

8.1.6 Rendering Equation

The Rendering equation provides a complete model for light interactions at both surfaces and within the medium of an object. It is used to calculate the radiance leaving a point on the object in a specific direction by integrating the incoming radiance from all directions. By solving the rendering equation, a path tracer can accurately simulate the light transport in a scene, including the effects of subsurface scattering. The function in its general form is given by [3]:

$$L_o(x_o, \vec{\omega}_o) = L_e(x_o, \vec{\omega}_o) + \int_{2\pi} f_s(x_o, \vec{\omega}_i, \vec{\omega}_o) L_i(x_i, \vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i)\, d\vec{\omega}_i \tag{20}$$

where:

- $L_o(x_o, \vec{\omega}_o)$ is the radiance leaving the object at point x_o in direction $\vec{\omega}_o$.
- $L_e(x_o, \vec{\omega}_o)$ is the radiance emitted by the object at point x_o in direction $\vec{\omega}_o$.
- $L_i(x_i, \vec{\omega}_i)$ is the radiance incident on the object at point x_i in direction $\vec{\omega}_i$, evaluated using the introduced RTE.
- $f_s(x_o, \vec{\omega}_i, \vec{\omega}_o)$ is the Bidirectional Scattering Distribution Function (BSDF) which describes the scattering properties of the object.

8.1.7 BSSRDF Model

While it is possible to accurately simulate subsurface scattering in a scene by solving the full Radiative Transfer Equation (RTE) and using the Bidirectional Scattering Distribution Function (BSDF) model, it is extremely computationally expensive (discussed in Section 8.2). The BSSRDF model allows for a more efficient approximation of the subsurface scattering effect by describing the light transport between two points in a participating media. The rendering equation can be extended to include the BSSRDF model as proposed by Jensen et al.[5]:

$$L_o(x_o, \vec{\omega}_o) = \int_A \int_{2\pi} S(x_i, \vec{\omega}_i, x_o, \vec{\omega}_o) L_i(x_i, \vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i)\, d\vec{\omega}_i\, dA(x_i) \tag{21}$$

where:

- $S(x_i, \vec{\omega}_i, x_o, \vec{\omega}_o)$ is the Bidirectional Scattering-Surface Reflectance Distribution Function (BSSRDF) which describes the light transport between two points in a participating medium.
- A is the incident area on the object.
- The L_e term is omitted as the object is not expected to emit light.

Assuming an idealized case of a ray of light carrying incident radiance at only one point on the object, the integrals in the extended rendering equation can be simplified, resulting in the following equation:

$$L_o(x_o, \vec{\omega}_o) = S(x, \vec{\omega}, x_o, \vec{\omega}_o) L(x, \vec{\omega}) \tag{22}$$

The BSSRDF function used here describes the light transport between two points x_i and x_o with incident and outgoing directions ω_i and ω_o respectively is defined as [5]:

$$S(x_i, \omega_i, x_o, \omega_o) = S_d(x_i, \omega_i, x_o, \omega_o) + S^{(1)}(x_i, \omega_i, x_o, \omega_o) \tag{23}$$

where:

- S_d is the diffuse component of the BSSRDF based on the observation that light tends to distribute isotropically in highly scattering media representing the light that is scattered multiple times within the object before exiting [25].
- $S^{(1)}$ - single scattering component accounting for the light that is scattered only once within the object before exiting.

8.1.8 Diffusion approximation

To approximate the multiple scattering component of the BSSRDF using diffuse approximation, standard dipole model is used. This model employs two virtual point sources to approximate this complex light scattering:

- S^+ - light source beneath the surface of the object respresenting the light entering the object and scattering within it.
- S^- - light source above the surface of the object, modeling light attenuation near the surface of the object.

this setup is illustrated in Figure 32. The positive light source is positioned beneath the surface at a distance z_r and the negative light source is positioned above the surface at a distance z_v. Positions can be obtained by solving the following equations[5]:

$$z_r = \frac{1}{\sigma_t'} \tag{24}$$

$$z_v = z_r + 4AD \tag{25}$$

where σ_t' is the reduced extinction coefficient, given by

$$\sigma_t' = \sigma_s' + \sigma_a \tag{26}$$

σ_s' is the reduced scattering coefficient:

$$\sigma_s' = \sigma_s(1-g) \tag{27}$$

D is the diffusion constant, calculated as:

$$D = \frac{1}{3(\sigma_t')} \tag{28}$$

A is the corrective term used for the Fresnel boundary conditions at an interface between mediums and is given by:

$$A = \frac{1+F_{dr}}{1-F_{dr}} \tag{29}$$

F_{dr} is the diffuse Fresnel reflectance term which accounts for the reflection of light at the surface of the object. It is approximated by the following equation:

$$F_{dr} = \frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta \tag{30}$$

where η is the index of refraction of the object.

Given the positions of the virtual light sources, the diffuse reflectance of the object can be calculated as [5]:

$$R_d = \frac{\alpha'}{4\pi}\left[(\sigma_{tr}d_r+1)\frac{e^{-\sigma_{tr}d_r}}{\sigma_t' d_r^3} + z_v(\sigma_{tr}d_v+1)\frac{e^{-\sigma_{tr}d_v}}{\sigma_t' d_v^3}\right] \tag{31}$$

The reduced albedo α' can be obtained by:

$$\alpha' = \frac{\sigma_s'}{\sigma_t'} \tag{32}$$

The transport coefficient σ_{tr}:

$$\sigma_{tr} = \sqrt{3\sigma_a\sigma_t'} \tag{33}$$

The distances d_r and d_v are the distances between the virtual light sources and the point of interest x:

$$d_r = ||x - x_r|| \tag{34}$$

$$d_v = ||x - x_v|| \tag{35}$$

Finally, by adding the Fresnel reflection at the boundary for incoming and outgoing directions of the light, the diffusion approximation of the BSSRDF representing multiple scattering can be calculated as:

$$S_d(x_i, \omega_i, x_o, \omega_o) = \frac{1}{\pi} F_t(\eta, \omega_i) F_t(\eta, \omega_o) R_d(||x_i - x_o||) \tag{36}$$

8.1.9 Single scattering

The single scattering term accounts for light that penetrates the surface of a material, scatters once inside, and then exits the materials at a different point. This term can be computed assuming a uniformly lit, homogeneous object. this calculation, however assumes a semi-infinite - object is thick enough to to not reach the other side of the object where the light would reflect back. The contribution of the single scattering adds sharp features to the subsurface scattering effect due to the light scattering only once within the object. The single scattering term is given by [5]:

$$S^{(1)}(x_i, \omega_i, x_o, \omega_o) = \alpha F \frac{p(\omega_i, \omega_o)}{|\vec{n} \cdot \omega_i| + |\vec{n} \cdot \omega_o|} \tag{37}$$

where:

- α is the albedo
- F is the Fresnel term as a product of Fresnel transmission terms for incoming and outgoing directions of the light
- p(ω_i, ω_o) is the phase function

8.1.10 Phase Function

A phase function is a component used in a various scattering models to describe the angular distribution of light scattered from surfaces. This is usually a 1D function of the angle θ between the incoming and outgoing directions of the light. The phase function is normalized such that the integral of the function over all directions is equal to 1.

One widely used phase function is the Henyey-Greenstein phase function which is defined as:

$$p_{HG}(\cos\theta) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}} \tag{38}$$

where g is the asymmetry parameter which determines the scattering direction. For $g = 0$, the phase function is isotropic, meaning that the light is scattered uniformly in all directions. For $g > 0$, the phase function is forward scattering, meaning that the light is scattered more in the forward direction. For $g < 0$, the phase function is backward scattering with light scattered more in the backward direction.

The visual effect of this is illustrated in Figure 22 for a rendered medium of a spherical shape. In this illustration, the left image shows the object with stronger backward scattering

and the right image exhibits the object with forward scattering. Since the light is coming from the back of the object, forward scattering leads to more light reaching the camera.

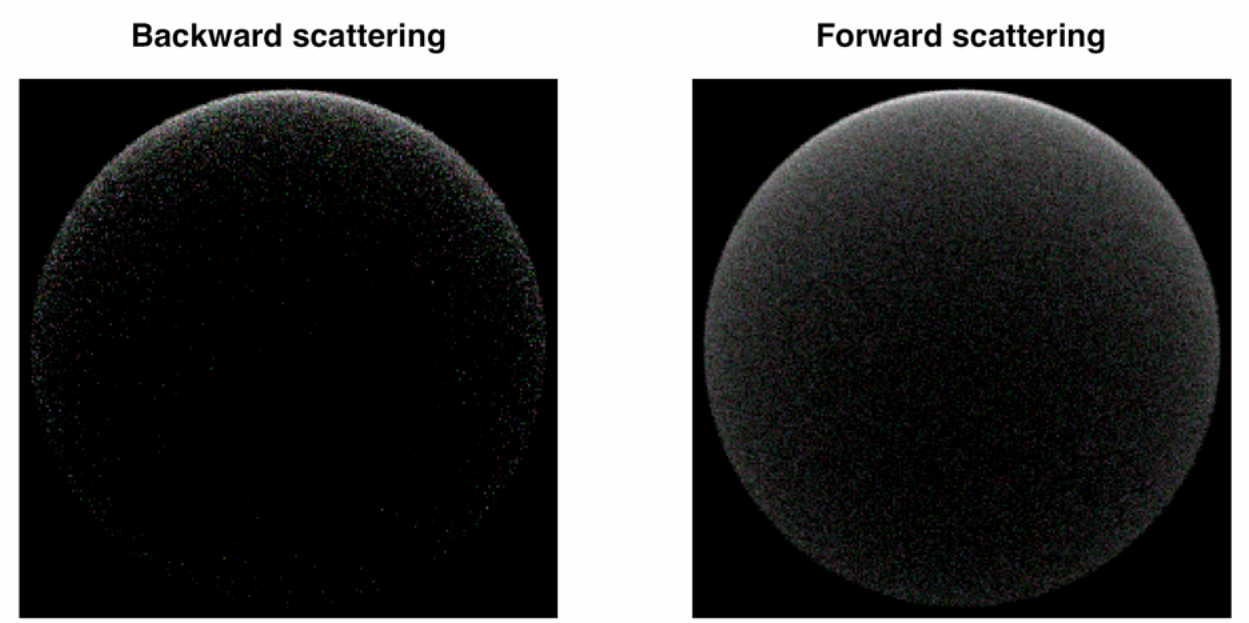


Figure 22: Comparison of a PBRT rendered medium illuminated from the opposite side of the camera with (left) stronger backward scattering (g = -0.5) and (right) stronger forward scattering (g = 0.5). Exposure of the images adjusted to 38% and -230% respectively.

8.1.11 Refraction

Snell's Law describes the refraction of light at the interface between two media with different refraction coefficient η. The law states that the ratio of the sines of the angles of incidence and refraction are equal to the ratio of the refractive indices of the two media. The law can be expressed as:

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\eta_2}{\eta_1} \tag{39}$$

In practice, this allows for the calculation of the direction of the refracted or incident ray when the index of refraction of the two mediums is known as well as for the further calculation of the Fresnel reflection and transmission terms used in the BSSRDF model (Section 8.1.7).

Fresnel reflection term gives the ratio of reflected light to the incident light at the boundary of the object. It is evaluated by first calculating the reflectances s-polarized and p-polarized light using the Fresnel equations [26]:

$$R_s = \left| \frac{\eta_1 \cos\theta_i - \eta_2 \cos\theta_t}{\eta_1 \cos\theta_i + \eta_2 \cos\theta_t} \right|^2 \tag{40}$$

$$R_p = \left| \frac{\eta_1 \cos\theta_t - \eta_2 \cos\theta_i}{\eta_1 \cos\theta_t + \eta_2 \cos\theta_i} \right|^2 \tag{41}$$

where θ_i and θ_t are the angles of incidence and refraction.

From there, the Fresnel reflection under the assumption that equal amount of power is in the s and p polarizations, the Fresnel reflection term can be evaluated as follows:

$$F_r = \frac{1}{2}(R_s + R_p) \tag{42}$$

and the Fresnel transmission term is then given by:

$$F_t = 1 - F_r \tag{43}$$

8.2 PBRT

This section is dedicated to describing the implementation of subsurface scattering in PBRT-v4. It will cover the specifics of the BSSRDF model used in PBRT, the material properties required for volumetric rendering of the object.

While it is possible to accurately simulate subsurface scattering in a scene by solving the full Radiative Transfer Equation (RTE) using PBRT-v4, it is extremely computationally expensive. This can be done by setting up the material properties of an object as a *Dielectric*, which involves assigning an index of refraction and roughness for surface light interactions and defining a medium within the object with specific scattering, absorption, and asymmetry parameters for internal light interactions.

The computational expense arises from the numerous rays that must be traced through the scene, accounting for internal interactions within the medium and bouncing multiple times within the object before exiting. For example, rendering a scene with these settings resulted in a render time of 1975 seconds, compared to 422 seconds using the *subsurface* material definition in PBRT (explored in Section 8.2.1). Additionally, the dielectric + medium approach produced significantly noisier results, as shown in the side-by-side comparison in Figure 23.

The subsurface scattering effect in the dielectric + medium image is less pronounced than in the output using *subsurface* material implementation, likely due to the need for a higher number of samples and ray bounces to achieve a similar effect, further increasing the render time.

Therefore, while fully rendering subsurface scattering by accounting for each internal light interaction within the object is an option, it yields unfavorable visual results for a short render time and is not practical given the hardware and time constraints. Consequently, this method is not used for comparing the subsurface scattering effect in PBRT.

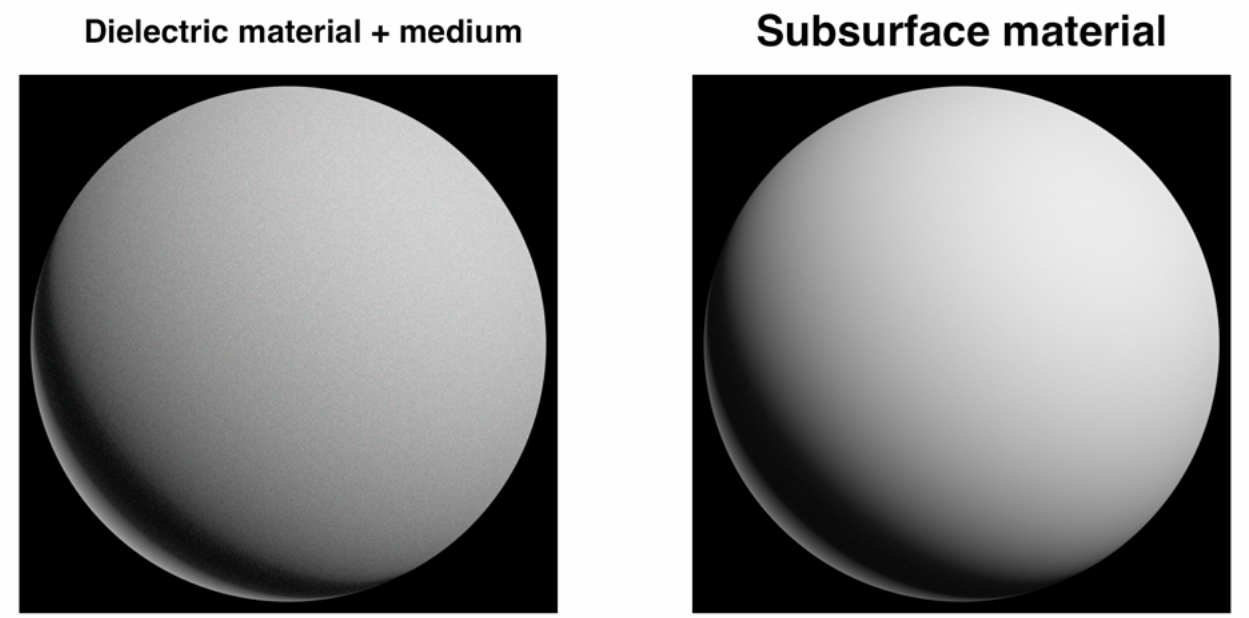


Figure 23: Comparison of subsurface scattering effect in PBRT using a dielectric object with a medium within it (left) and using a subsurface material definition (right). Exposure of the images adjusted to -92% and -250% respectively.

8.2.1 Material

Delving into the implementation of *subsurface* material in that will be used for rendering images for the comparison, the material is defined by the parameters discussed in Section 8.1 - parameters required for volumetric rendering of the object. The material is defined by the following parameters:

- σ_a - absorption coefficient
- σ_s - scattering coefficient (typically given as reduced scattering coefficient)
- η - index of refraction
- g - asymmetry parameter
- scale - scale factor applied to the scattering and absorption coefficients to adjust the optical thickness of the object

Absorption and scattering coefficients are expressed in per unit distance measures, so it is important to scale these coefficients according to the portrayed thickness of the object.

Additionally *subsurface* material can have its surface roughness specified by roughness parameter affecting the surface reflection of the object.

The material can also be defined using a *reflectance* and *mfp* (mean free path of light) parameters to eliminate the need to specify the absorption and scattering coefficients, However, this approach was not explored or used in this work. PBRT also provides measured

scattering properties for different materials such as skin, milk, marble that can be referenced by its name when defining the material [22].

8.2.2 Subsurface scattering in PBRT

The implementation of subsurface scattering when using the *subsurface* material in PBRT leverages the BSSRDF model. This specific approximation uses a photon beam diffusion (PBD) technique introduced by Habel et al. [27] to refine the calculations for the single scattering and multiple scattering components of the BSSRDF. PBD interprets incident light as a beam of photons. For a large number of photons, going through the medium, the PBD technique calculates radiances at these sample points and averages them to create a more accurate diffusion profile. This data is then stored in a table and used to evaluate the BSSRDF at runtime, saving computational resources by avoiding the need to recalculate the BSSRDF for each ray.

8.2.3 Diffusion approximation

The diffusion approximation used to calculate the multiple scattering component of the BSSRDF, employ the dipole solution as discussed in Section 8.1.7. Enhancements for this model were made to utilize a modified diffusion coefficient D, developed by Grosjean [28], which is given by:

$$D = \frac{2\sigma_a + \sigma_s'}{3(\sigma_a + \sigma_s')^2} \tag{44}$$

This change aims for superior accuracy when compared to classical diffusion approximation.

PBRT's dipole approximation also improves approximation of boundary conditions for the dipole model by adjusting the placement of the virtual light source by leveraging the concept of extrapolation depth. This is done by calculating the extrapolation depth z_e as [3]:

$$z_e = -2D\frac{1 + 3F_{r,2}(\eta)}{1 - 2F_{r,1}(\eta)} \tag{45}$$

where $F_{r,1}$ and $F_{r,2}$ are the Fresnel moments accounting for the reflection of light at the boundary of the object (explained in section 8.3.1). The extrapolation depth is then used to calculate the depth of the virtual light source as:

$$z_v = 2z_e - z_r \tag{46}$$

Given the improved diffusion coefficient and boundary conditions, diffuse radiant exitance of the object is evaluated using the improved BSSRDF model by d'Eon and Irving[29]:

$$E = C_\phi(\eta)\Phi_d - C_e(\eta)E_D \tag{47}$$

By scaling E term by reduced albedo α' and a correction factor κ, the multiple scattering component S_d of the BSSRDF is calculated as [30]:

$$S_d(x_i, \omega_i, x_o, \omega_o) = \alpha'^2 E \kappa \tag{48}$$

The κ correction factor used to adjust for light source close to the surface of the object, explored in [27] and additional α' term used here accounts for the energy reduction due to repeated interraction within the medium [30]. While the resulting benefits from these improvements in approximation over Jensen's BSSRDF model are beyond the scope of this project, they are detailed in the PBRT-v3 documentation [30].

8.2.4 Single scattering

Examining the single scattering component of the BSSRDF, PBRT employs a single scattering term evaluated using monte carlo integration described by Habel et al. [27] computed from the following equation[30]:

$$S^{(1)}(x_i, \omega_i, x_o, \omega_o) = \alpha \frac{p(\omega_i, \omega_o) e^{-\sigma_t(d+t_{crit})} F \cos\theta_o}{d^2} \tag{49}$$

containing an attenuation term $e^{-\sigma_t(d+t_{crit})}$ which is influenced by the distance d between two points of the medium and the critical distance t_{crit} which is the distance at which the light is totally internally reflected at the boundary of the object is calculated as:

$$t_{crit} = r\sqrt{\eta^2 - 1} \tag{50}$$

8.3 OpenGL Model 1

Implementing subsurface scattering in a rasterizer such as OpenGL is a challenging task due to the nature of rasterization primarily designed to work on the surface of the object. Additionally, rasterization itself does not inherently support volumetric rendering, making it difficult to simulate any internal interactions of light within the object. To address this, Jensen et al. [5] introduced a method to approximate subsurface scattering with a Bidirectional Reflectance Distribution Function (BRDF) model under the assumption that the incident light is uniformly distributed across the object. Using this method the BSSRDF of semi-infinite medium can be calculated. Semi-infinite medium implies that the object is thick enough for the light to not reach the other side of the object where the light would reflect back. As described in the section 8.1.7, the BSSRDF model used in OpenGL is a combination of the diffusion approximation and single scattering components.

8.3.1 Diffusion approximation

Assuming the limiting conditions, the diffusion approximation can be used to calculate the multiple scattering component of the BSSRDF with the diffuse reflectance for a material calculated as follows:

$$R_d = \frac{\alpha'}{2}\left(1 + e^{-4/3A\sqrt{3(1-\alpha')}}\right) e^{-\sqrt{3(1-\alpha')}} \tag{51}$$

The corrective term in this equation, described in section 8.1.8, was replaced with a calculation from an improved BSSRDF model by d'Eon and Irving [29] for potentially improved results which is given by:

$$A = \frac{1 - C_E(\eta)}{2C_\phi(\eta)} \tag{52}$$

These two terms - C_E and C_ϕ are Fresnel moments that account for the reflection of light at the boundary of the object and can be expressed as:

$$C_E(\eta) = \frac{1}{2}(1 - 3C_2)C_\phi(\eta) = \frac{1}{4}(1 - 2C_1) \tag{53}$$

The terms C_1 and C_2 can be expressed as polynomial functions of the index of refraction η of the object:

$$2C_1 \approx \begin{cases} +0.919317 - 3.4793\eta + 6.75335\eta^2 - 7.80989\eta^3 + 4.98554\eta^4 - 1.36881\eta^5 & \eta < 1 \\ -9.23372 + 22.2272\eta - 20.9292\eta^2 + 10.2291\eta^3 - 2.54396\eta^4 + 0.254913\eta^5 & \eta \geq 1 \end{cases} \tag{54}$$

$$3C_2 \approx \begin{cases} +0.828421 - 2.62051\eta + 3.36231\eta^2 - 1.95284\eta^3 \\ \quad +0.236494\eta^4 + 0.145787\eta^5 & \eta < 1 \\ -1641.1 + \frac{135.926}{\eta^3} - \frac{656.175}{\eta^2} + \frac{1376.53}{\eta} \\ \quad +1213.67\eta - 568.556\eta^2 + 164.798\eta^3 \\ \quad -27.0181\eta^4 + 1.91826\eta^5 & \eta \geq 1 \end{cases} \tag{55}$$

Scaling the diffuse reflectance by two Fresnel tranmission terms for incoming and outgoing directions of the light, the diffusion term for BSSRDF is calculated as:

$$S_d(x_i, \omega_i, x_o, \omega_o) = \frac{1}{\pi} F_t(\eta, \omega_i) F_t(\eta, \omega_o) R_d \tag{56}$$

8.3.2 Diffuse reflectance Evaluation

Correctness of the implementation of diffuse reflectance calculations in OpenGL described in section 8.3.1 was verified using a measured parameter table from Jensen et al. [5]. Table 1 contains the material properties of a certain material assuming a refractive index of 1.3 and its corresponding diffuse reflectance values for red, green, and blue color channels. The material properties were used to calculate the diffuse reflectance for the material in OpenGL and the results were compared to the values in the table to avoid potential errors in the implementation.

Material	σ_s' [mm^{-1}]			σ_a [mm^{-1}]			Diffuse Reflectance			η
	R	G	B	R	G	B	R	G	B	
Apple	2.29	2.39	1.97	0.0030	0.0034	0.046	0.85	0.84	0.53	1.3
Chicken1	0.15	0.21	0.38	0.015	0.077	0.19	0.31	0.15	0.10	1.3
Chicken2	0.19	0.25	0.32	0.018	0.088	0.20	0.32	0.16	0.10	1.3

Table 1: Material properties and their corresponding diffuse reflectance values for red, green, and blue color channels [5].

8.3.3 Single scattering

As for single scattering term for the OpenGL implementation, it is calculated using the formula discussed in section 8.1.9 used in the original BSSRDF model 37. The asymmetry parameter, while determining the scattering direction, in the case of BSSRDF model presented as BRDF essentially just adjusts perceived radiance of the surface of an object. The phase function used in the single scattering term is the Henyey-Greenstein phase function as discussed in section 8.1.10.

8.3.4 Material properties

Considering the calculations for the diffusion approximation and single scattering components of the BSSRDF model, the material properties required for the OpenGL implementation are the same as those used to define a *subsurface* material in PBRT. These include:

- σ_a - absorption coefficient
- σ_s - scattering coefficient
- η - index of refraction
- g - asymmetry parameter

scaling factor applied to the scattering and absorption coefficients is of no use in OpenGL implementation as the thickness of the object is assumed to be infinite and does not affect the reduced albedo α' parameter which is the ratio of the reduced scattering coefficient to the reduced extinction coefficient 32.

8.4 Visual comparison

8.4.1 Scattering and absorption coefficients

The first comparison set between PBRT and OpenGL BSSRDF models was performed using the same scene described in Section 5, where the scale of the object was changed in PBRT (by scaling the scattering and absorption coefficients) - effectively changing the perceived optical depth of the material without altering its physical thickness. This allows to evaluate the BSSRDF model at different levels of transparency. In this case, since the optical

thickness of the object **cannot** be modified in **OpenGL** (scaling of absorption and scattering coefficients do not modify the ratio between them), scaling PBRT material properties was used to simulate a semi-infinite medium in PBRT. The full results of this comparison are shown in Figure 28. The comparison explores how the BSSRDF model in OpenGL approximates the subsurface scattering for materials with different ratios of scattering to absorption coefficients.

Initial visual inspection of the images shows that results regardless of the material properties used, seem to hold up to the PBRT results for objects through which light cannot penetrate deep enough to reach the other side. The most similar results between the two renderers are that of a high scaterring material hinting that such BSSRDF model in OpenGL could be used to simulate the subsurface scattering effect in such cases.

However, if the goal is to represent a material that is more transparent (as depicted by smaller scaling factors in PBRT), the results from OpenGL are not suitable for such cases due to limitations that BRDF model has in representing the BSSRDF model and is clearly seen in the difference images.

8.4.2 Refraction index

The second comparison (Figure 29 focuses on the influence of material's index of refraction on the accuracy of the BSSRDF model in OpenGL. Going off the results from Section 8.4.1, object was rendered with high scattering and quite low absorption material properties, to have a minimal error before modifying the refraction index of the material. Unfortunately, the results when using the same material properties were not as accurate as expected, with differences being more apparent in terms of radiance of the object as the refraction index is increased.

Further reducing the absorption coefficient to 0.001 repeating the comparison shows a significant visual improvement in the results being visually indistinguishable from images from PBRT.

8.5 Quantitative comparison

In this section, the RSE errors between the images rendered in PBRT and OpenGL are analyzed, focusing on causes behind the errors seen in different comparisons of the BSSRDF models in OpenGL and PBRT.

8.5.1 Single scattering term

Looking at the RSE error images from the comparisons, highly scattering material images show a similar pattern of errors across the image. This is the result of single scattering term of BSSRDF. Single scattering term in PBRT is approximated over the Area of incidence, while in OpenGL it is only calculated per point of the object using an approximation for a semi-infinite medium as presented in Section 8.1.9. The single scattering term from PBRT is more uniform across the object, which can be seen from Figure 24 where the RSE image

of the object rendered in OpenGL without single scattering term is shown. The remaining error is then uniform across the lit surface of the object.

Figure 24: RSE image of the object rendered with OpenGL without the single scattering term.

8.5.2 Asymmetry parameter

To briefly address the asymmetry parameter, for semi-infinite medium, the asymmetry parameter does not have a significant effect as opposed to what is portrayed by Section 8.1.10. g parameter for a very optically thick object controls the intensity of single scattering term and therefore the parameter was not explored further.

8.6 Error analysis

Images showing RSE between the images rendered in PBRT and OpenGL for this comparison was not filtered, but instead just cut around the object to remove the edge artifacts (described in Section 6.3.1). As with previous comparisons, the rendering results have their exposure adjusted to accommodate the brightest images. Some images showing the differences between the two rendered images have individual exposure values stated under each image.

To explore how optical thickness and different ratios of scattering to absorption coefficients affect the accuracy of the inspected BSSRDF model in OpenGL (from Figure 28), the RSE images were analyzed. Graphs in Figure 25 illustrate the magnitudes of RSE when OpenGL implementation is compared to PBRT's with different optical thicknesses. The results align with the visual comparison, verifying that the BSSRDF model in OpenGL is more accurate for highly scattering opaque materials with minimal absorption. Futhermore, optically scaling the material properties in PBRT beyond a certain point causes no change in visual or RSE results.

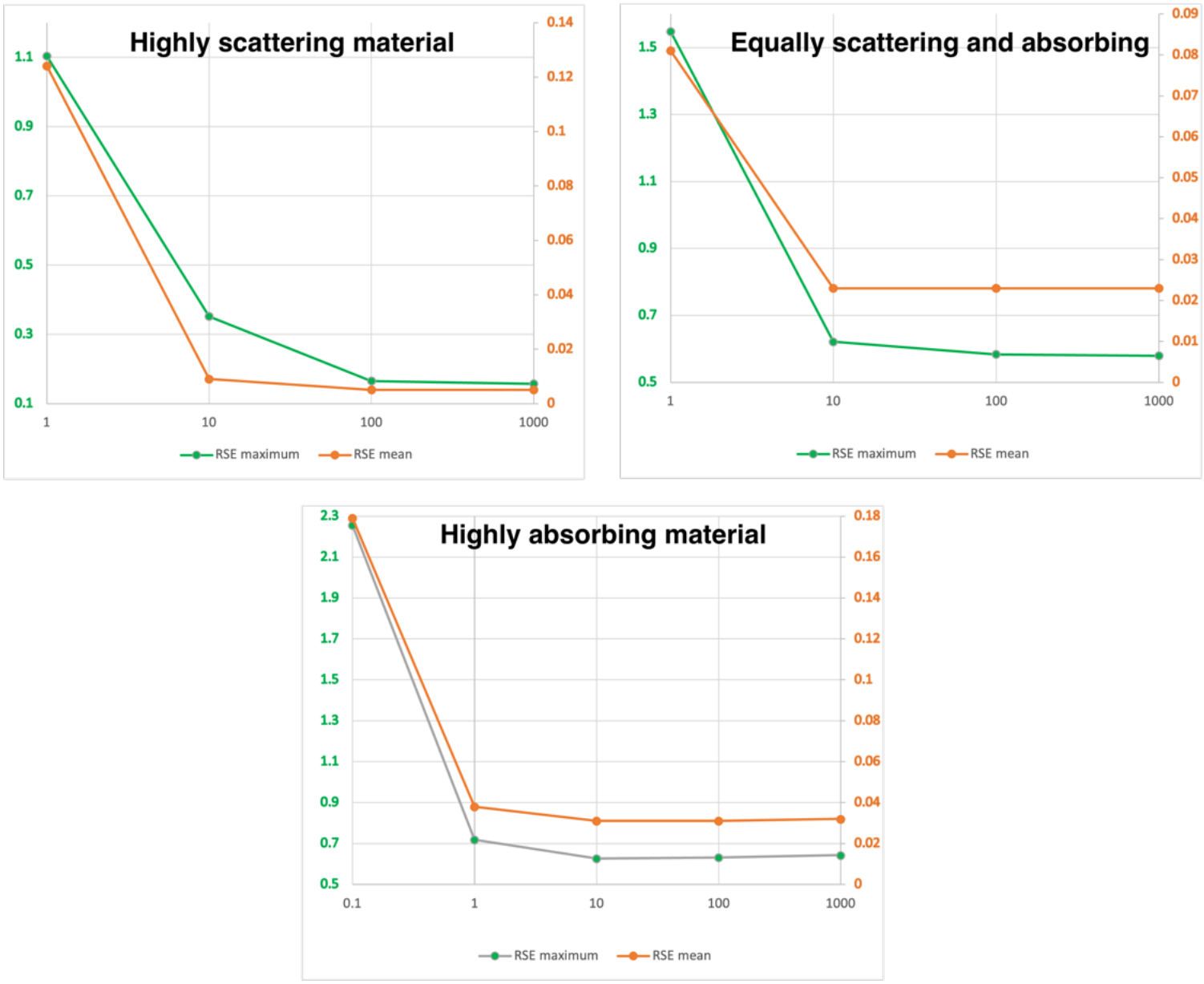


Figure 25: Graphs visualising RSE change for 3 materials with different scattering to absorptions ratios and varying optical thickness.

Delving into the error data for images of the object with different refraction indices (Figure 29), the same high scattering and low absorption material used in the previous comparison. This resulted in exponentially increasing RSE values with the increase of the refraction index of the material (as seen in Figure 26). To investigate this further, another material with absorption coefficient reduced to 0.001 was used to repeat the comparison. The results showed a significant improvement in the magnitude of RSE values as well as the increase with the refraction index of the material. This solidifies the observation that the BSSRDF model in OpenGL is more accurate for highly scattering materials with min-

imal absorption. Using such higher absorption materials with this model can still produce convincing results, but should be avoided for higher refraction indices.

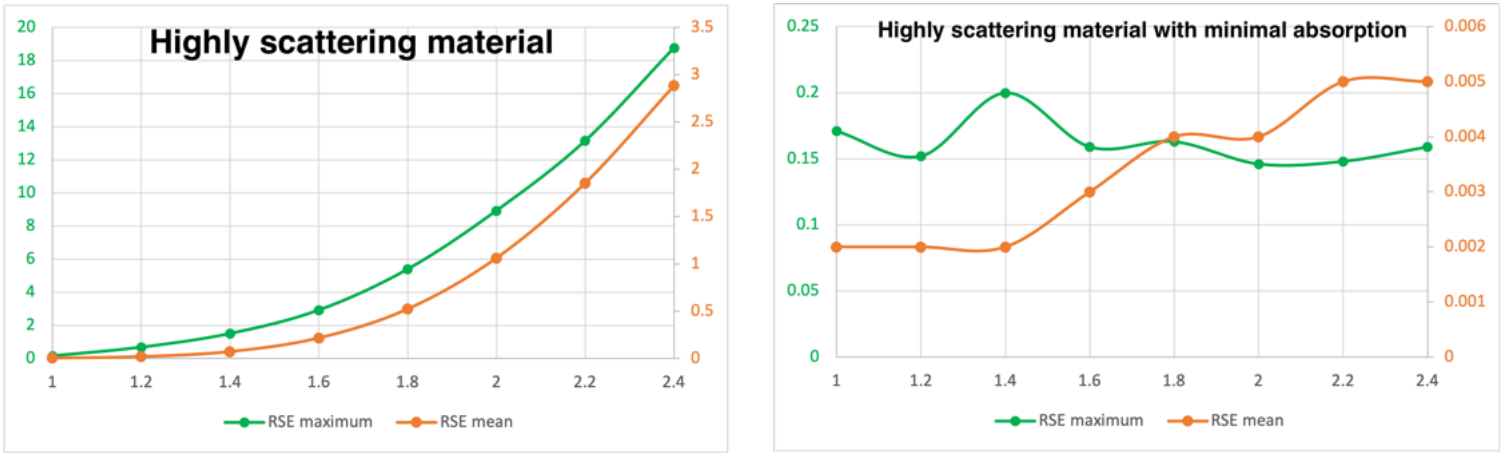


Figure 26: Graphs visualising RSE change with different refraction indices for two materials with different absorption coefficients.

Different error amount depending on the refraction index as well as the scattering to absorption ratio also means that if an actual material is to be represented with different scattering and absorption properties for different color channels, using this BSSRDF model in OpenGL could cause a shift in color balance of the object. To illustrate this, an apple material from the measured parameter table was used and rendered both with PBRT and OpenGL. This is visualised in Figure 27.

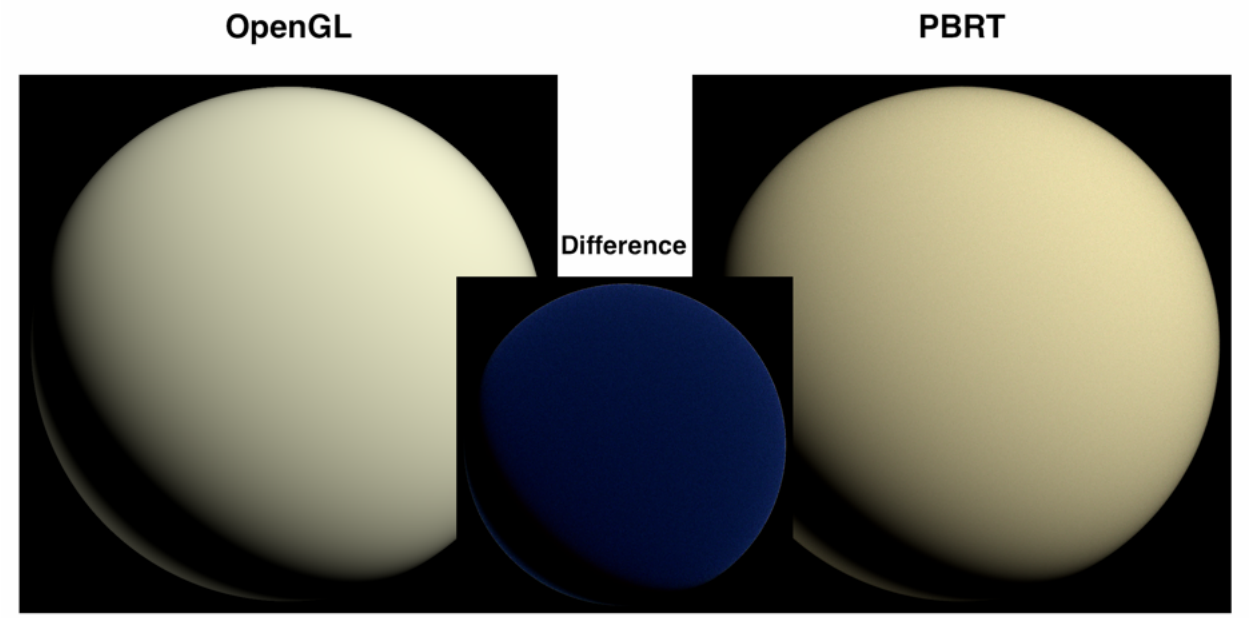
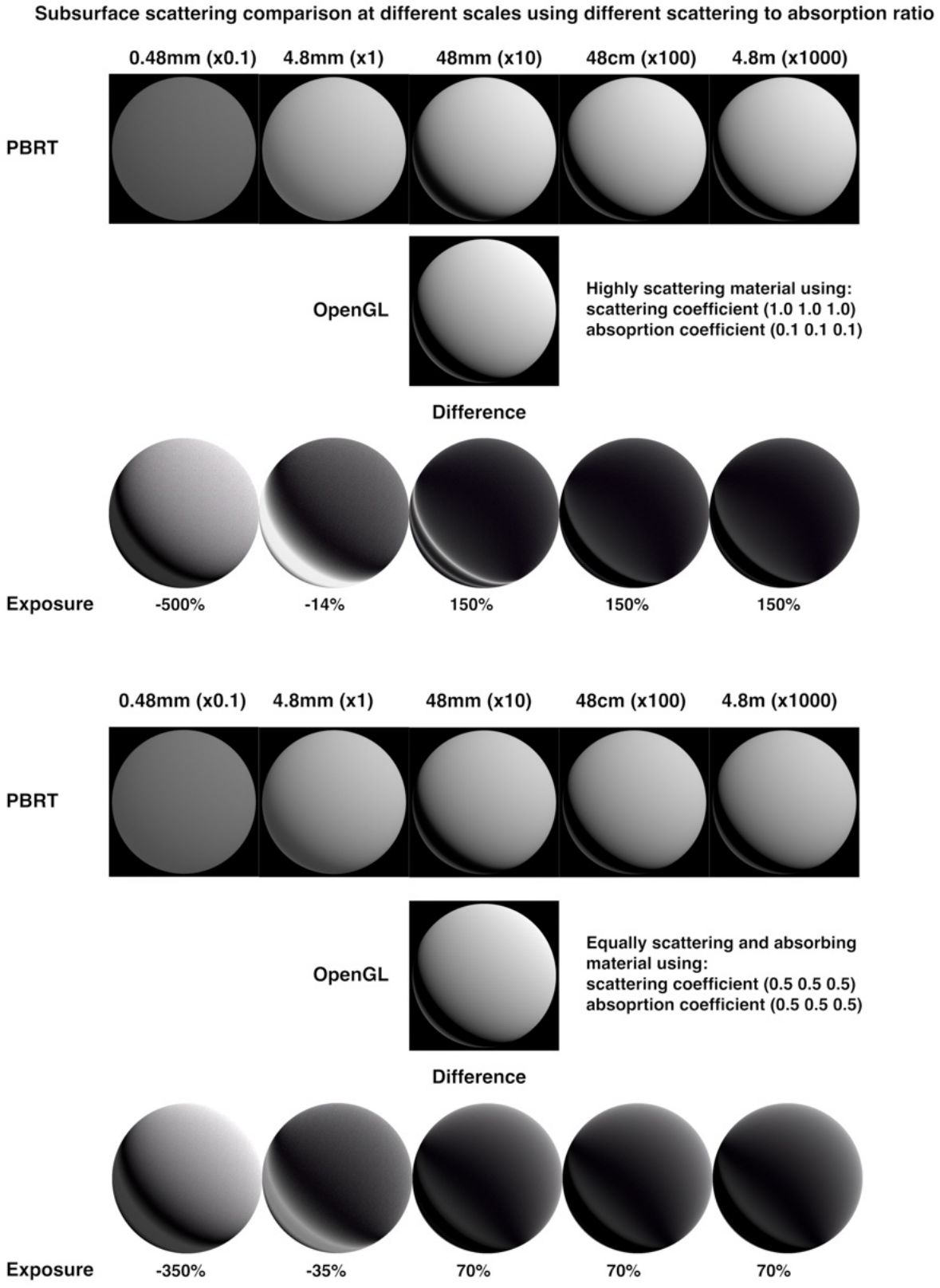


Figure 27: Comparison of the Apple material rendered in PBRT and OpenGL with refractive indexof 1.6, highlighting the color shift in the PBRT image.


Subsurface scattering comparison at different scales using different scattering to absorption ratio
0.48mm (x0.1)
4.8mm (x1)
48mm (x10)
48cm (x100)
4.8m (x1000)
PBRT
OpenGL
Highly scattering material using:
scattering coefficient (1.0 1.0 1.0)
absoprtion coefficient (0.1 0.1 0.1)
Difference
Exposure
-500%
-14%
150%
150%
150%
0.48mm (x0.1)
4.8mm (x1)
48mm (x10)
48cm (x100)
4.8m (x1000)
PBRT
OpenGL
Equally scattering and absorbing
material using:
scattering coefficient (0.5 0.5 0.5)
absoprtion coefficient (0.5 0.5 0.5)
Difference
Exposure
-350%
-35%
70%
70%
70%

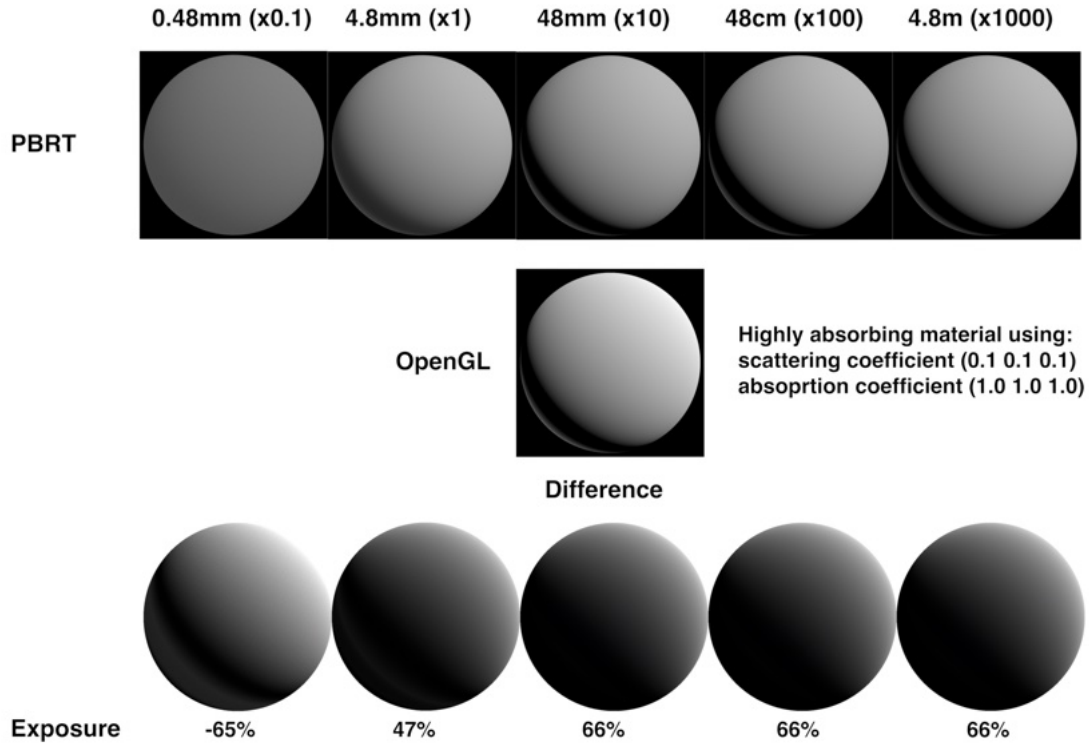


Figure 28: Rendered images of the object using different material properties in PBRT and OpenGL and their corresponding difference images showing the RSE values between them. Material properties in PBRT were scaled to modify optical thickness. Highly scattering material images (first comparison) have exposure adjusted to -196%. Equally scattering and absorbing material images (second comparison) have exposure adjusted to -38%. Highly absorbing material images (third comparison) have exposure adjusted to 232%. Difference images have their exposure values stated under each image.

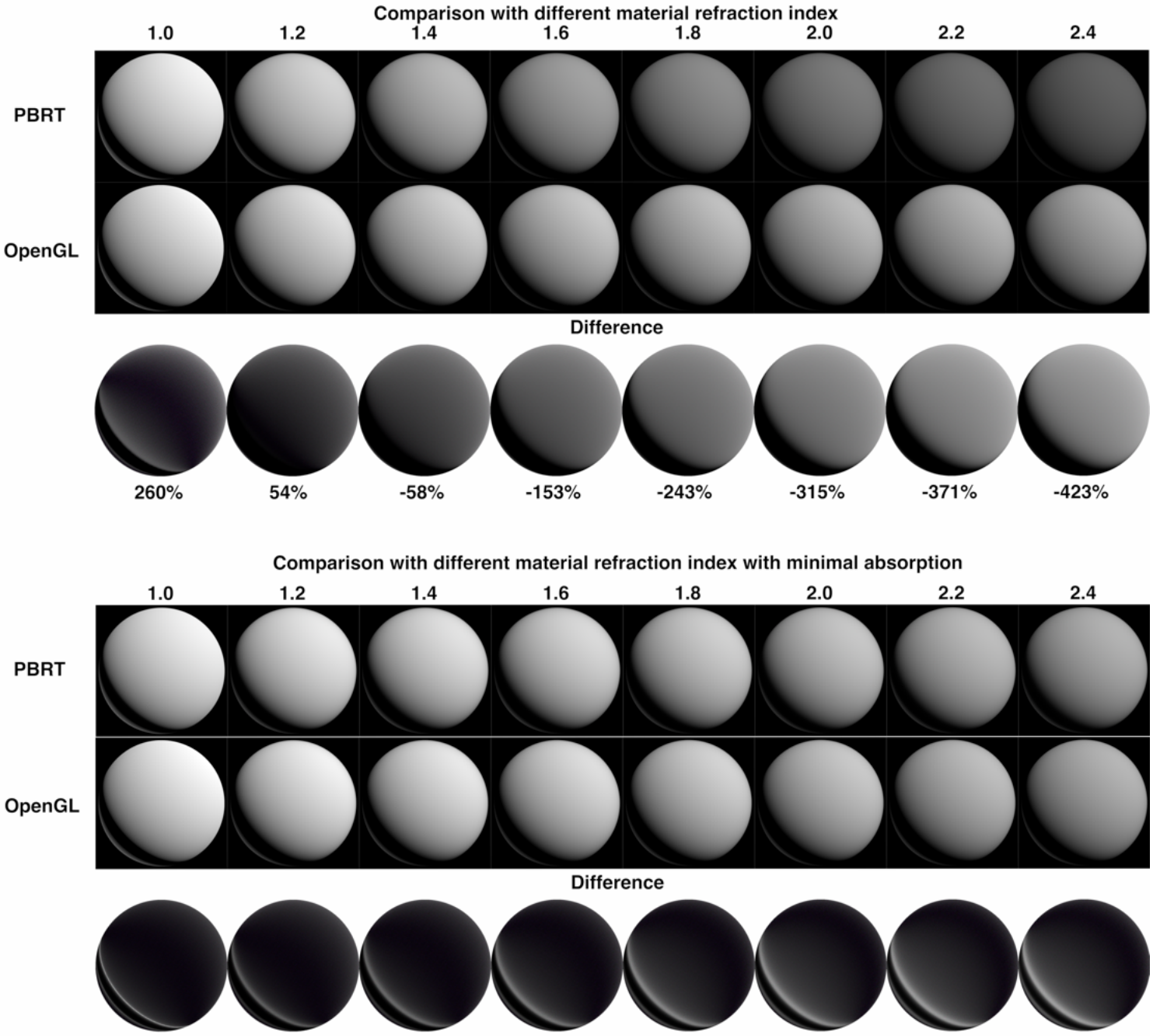


Figure 29: Rendered images of the object with varying refraction indices in PBRT and OpenGL, along with difference images showing the RSE values. For the top comparison, PBRT material properties were scaled for a semi-infinite medium, using highly scattering properties as detailed in Figure 28. For the second comparison, the absorption coefficient was further reduced to 0.001. The exposure settings for the rendered images adjusted to -180% (first comparison) and -280% (second). Difference images have exposure values of 230% or are labeled below each image.

8.7 OpenGL Model 2

To improve the effect of subsurface scattering in OpenGL, the dipole method was employed, as described in section 8.1.7. This technique allows for a more realistic simulation of light interacting with translucent materials by approximating the volumetric distribution of light sources. With the dipole method, the scene rendered in OpenGL can simulate light placement on the opposite side of the camera, thereby achieving a more accurate scattering of light within the object's volume and not being limited to presenting an opaque material. Initially, the primary challenge in using this method was the lack of thickness data for the object, which is needed for calculating the distances between the two point sources and the point of interest. By addressing this limitation, the implementation of the dipole method enhanced the visual effect of subsurface scattering in rendered scenes.

8.7.1 Thickness estimation

While the thickness (used for entry and exit point distance calculations for difussion approximation) of the object can be assumed for an object such as the one used for renderer comparison - the sphere, implying the thickness of the object would not be as simple for more complex geometry and therefore required some sort of real-time method to estimate it.

Idea of using the depth buffer to estimate the thickness of the object was explored. Depth buffer in a rasterizer like OpenGL is used to store the depth of each pixel in the scene to determine the visibility of objects in the scene. Using concepts of shadow mapping [31], scene was rendered from the perspective of the light source to create a depth map of the rendered object. This is done with the use of a framebuffer object (FBO) which allows to render to a different target than the default screen buffer. Attaching a depth texture to it, it is possible to render the scene from the perspective of the light source and store the depth values of the object in the texture. This depth texture can then be used in a shader to calculate the thickness of the object.

To perform this calculation, the depth texture needs to be correctly sampled in the shader, therefore the light-space coordinates of the fragment need to be calculated. This is done by transforming the fragment's position to the light's view space by multiplying it by the light's view-projection matrix. So if in a screen rendering pass a fragment is evaluated that is at the back of the object from the light's perspective, transforming it to the light's view space would result in a texture coordinate that corresponds to a depth value of a fragment in the front face of the object. Thickness can then be calculated by subtracting the front depth value from the back depth value.

Lastly, it is important to note that the depth data stored in a depth buffer is not linear in the given clips space with values varying exponentially with distance from the camera, therefore the depth values need to be linearized to get a subsequently valid thickness value. This can be done by using the following formula [31]:

$$linearDepth = \frac{2 * zNear * zFar}{zFar + zNear - depth(zFar - zNear)} \tag{57}$$

where $zNear$ and $zFar$ are the near and far planes of the camera and depth is the depth

value transformed to range [-1, 1].

This estimated thickness value can then be used in the shader to calculate the distances between the two virtual light sources and the point of interest to perform the dipole method calculations. This is illustrated by a Figure 30, where darker values represent smaller depth values and lighter values represent larger depth values.

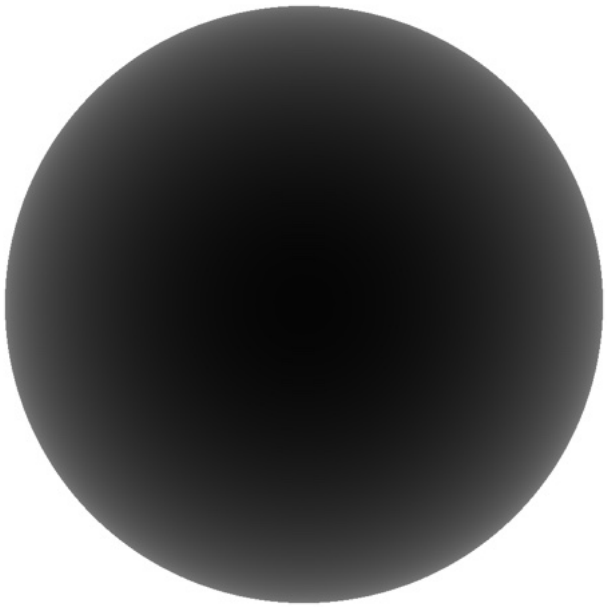


Figure 30: Depth data for the area of the object facing the light source.

Unfortunately, by using this method, the thickness estimation is not very accurate and requires setting near and far planes of the camera to be as close to the object as possible to get a more accurate thickness estimation. This method was not used in the final implementation of the dipole method in OpenGL due to the inaccuracy of the thickness estimation. Instead, by leveraging the same concept, instead of storing the depth data in the texture, the fragment coordinates were stored, so the thickness could be calculated by calculating the length of the vector between the front and back face of the object. By using this data, the thickness estimation was more accurate without having to constrain the camera's near and far planes.

It should be noted that since the scene used for all comparisons described in Section 5 uses multiple directional light sources, the fragment coordinates from the perspective of each light source were stored in separate textures to calculate the thickness from the perspective of each light source.

8.7.2 Dipole method implementation

Acquired thickness data allows to implement a more accurate diffuse approximation of the BSSRDF model in OpenGL to approximate a multiscattering within the object so that the object is not limited to representing the object as a semi-infinite medium. The dipole method was implemented in the fragment shader to calculate the diffuse reflectance part of the diffusion approximation used in BSSRDF as it is described in Section 8.1.8 using Equation 31. The thickness data was used to calculate the distances from the two point sources (virtual and a real one) and the point of interest to perform the dipole method calculations.

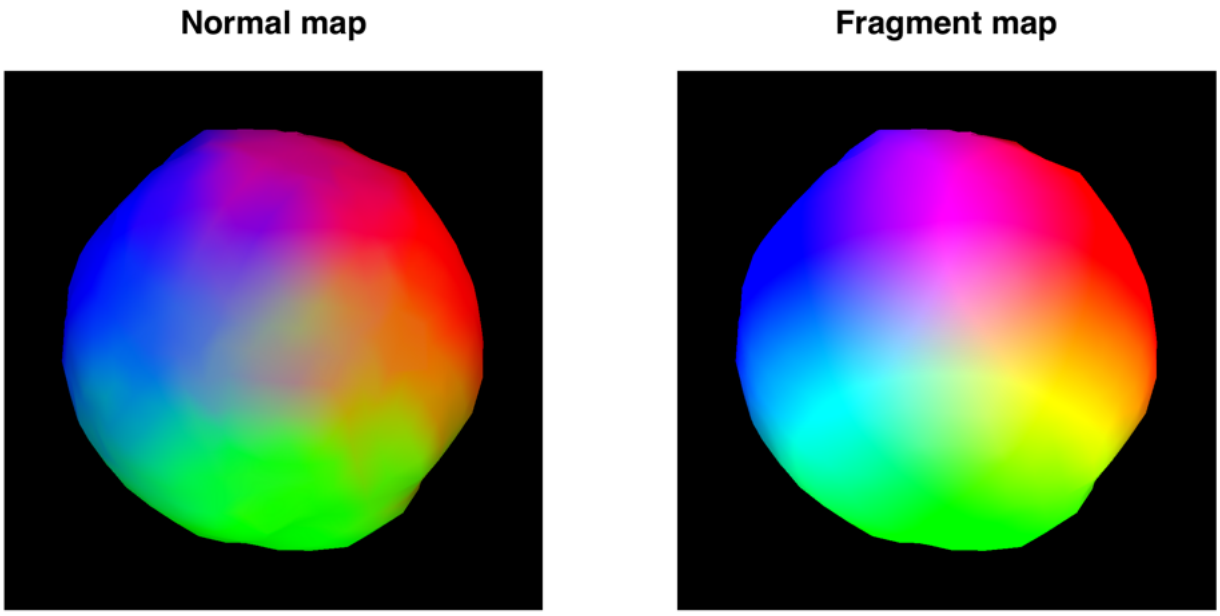


Figure 31: Fragment coordinates and normal vectors stored in textures for the dipole method implementation.

Using the dipole method changes the setup so that the point of incidence is no longer assumed to be the same as the light exit point, most notably, the fragments that do not have a direct line of sight to the light source are now taken into account. For this model it is assumed that on such fragment the light is incident from the opposite side of the object and by using the thickness data for the distance between these to fragments, the two light sources and the point of exit can be calculated to find the radiance of the object at that point. To correctly account for light incident on the entrance point of the object, it was necessary to obtain the normal vector data which was achieved using the same technique of storing the normal vectors from the perspective of the light source as described in Section 8.7.1. The overall setup of this implementation is illustrated in Figure 32 where radiance of x_o point is evaluated from a point x_i on the opposite side of the object.

The idea is to extend the Jensen's proposed BRDF model to account for light transfer in semi-infinite materials with an additional multiple scattering component using the dipole

method. Under the assumption that granular materials are usually round, the area of incidence was assumed to be a circle positioned at a centroid of a directly lit hemisphere. This basically implies the object to have a flat surface facing the light source. Given these constraints, the distance from the point of incidence to the point of exit was evaluated as follows:

$$d = \frac{3R}{8} + \frac{|x_o - x_i|}{2} \tag{58}$$

where R is the radius of the area of incidence and $\frac{3R}{8}$ is the distance from the centroid of the center of a sphere. The idea behind the model was to try to incorporate diffusion approximation with the use of a *single* sample point per fragment to evaluate the radiance of the object at the point of exit. The benefits of using the dipole method in this model are that it allows for the translucency to be defined using the material properties of the object instead of making completely arbitrary assumptions about the object.

The radiance at the point of exit was evaluated using the following equation:

$$L_o(x_o, \omega_o) = L_i(x_i, \omega_i)(A * S_{d2}(x_i, \omega_i, x_o, \omega_o)) + (S_d(x_i, \omega_i, x_o, \omega_o) + S^{(1)}(x_i, \omega_i, x_o, \omega_o)) \tag{59}$$

where:

- A - area of incidence
- S_{d2} - multiple scattering component using the dipole method
- S_d - multiple scattering component using the diffusion approximation from the first OpenGL model
- $S^{(1)}$ - single scattering term from the first OpenGL model

The old multiple scattering component was kept in the model to account for the multiple scattering term in the area of incidence, while the new multiple scattering component using the dipole method was used to account for the multiple scattering term on the back side (not facing the light source) of the object. The single scattering term was also kept in the model to account for the single scattering component of the BSSRDF model.

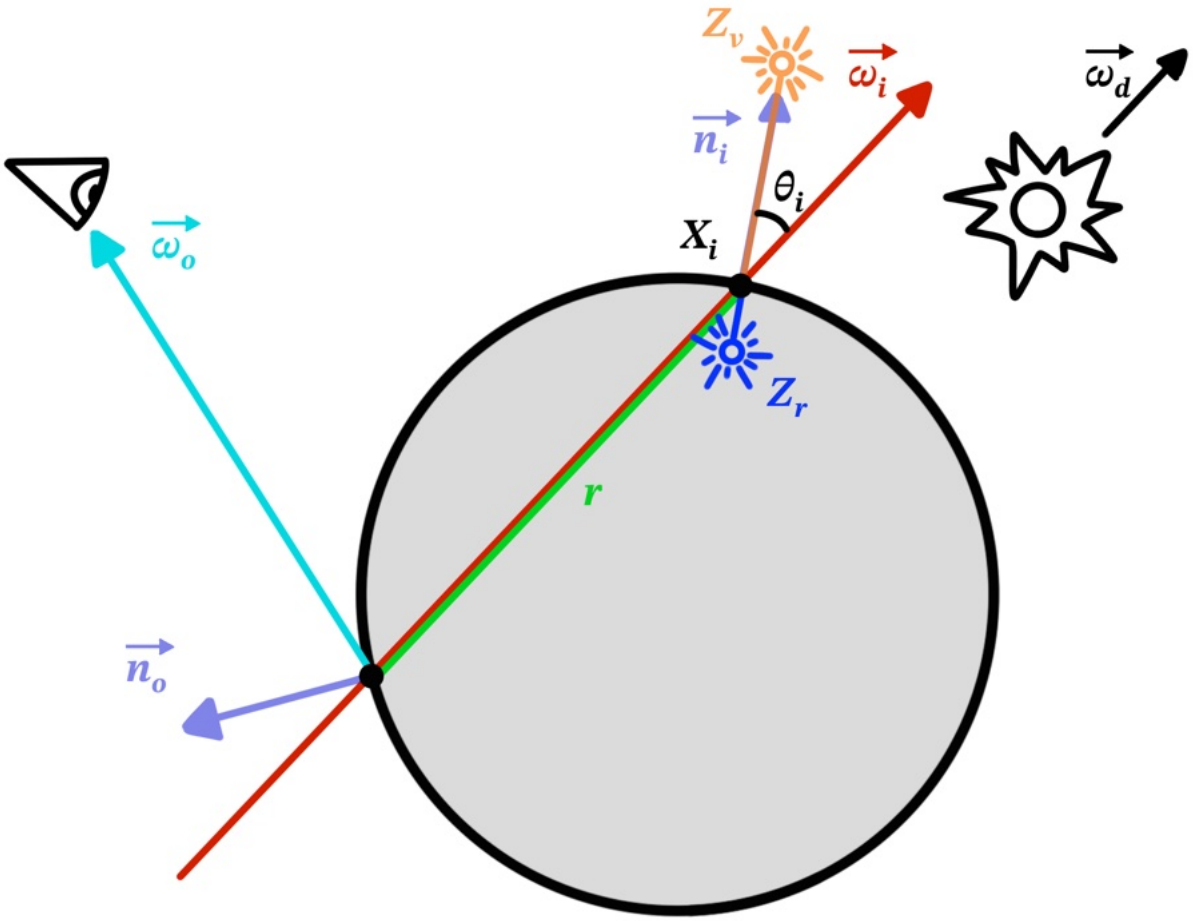


Figure 32: Dipole setup for diffusion approximation in OpenGL.

8.7.3 Visual Comparison

As this model allows for the translucency to get past the azimuthal line of incidence, to compare the results from this BSSRDF implementation in OpenGL to PBRT, it was decided to use a scene with a single directional light source at the top right of the object to make the effect of translucency of the object more pronounced while still making sure to keep it within the constraints of a semi-infinite medium. The results from this model were compared with the results from the first presented model using the same setup to see if there is improvement in visual and quantitative accuracy. Three different materials were used for the comparison - a highly scattering material, a material with equal scattering and absorption coefficients, and a highly absorbing material as illustrated by Figure 33. The renderings had to have their optical thickness adjusted in order to maintain the semi-infinite medium constraint.

Overlooking the results, the output from this model are under *Model 2* label of Figure 33 and the unfortunate part is that single scattering term has very clearly visible edges of incidence, which is there to account for very optically thick medium and tailored for use on a infinitely thick flat slab. It does not work visually well with the perceived translucency of the object. While there were no other known solutions (simple solutions) to approximate

the single scattering term more accurately, the term was kept with slight modification to the fall-off of the term to make it more coherent with the other side of the object - to portray the visual potential of the model in the results. The effect of translucency is noticeable in the OpenGL images under all three materials and from a visual perspective, manages to represent the object in a more realistic way than the first presented model. It does not look as natural as PBRT's especially for a high scattering material, but for a higher absorption material, the results certainly portray the object in a more realistic way.

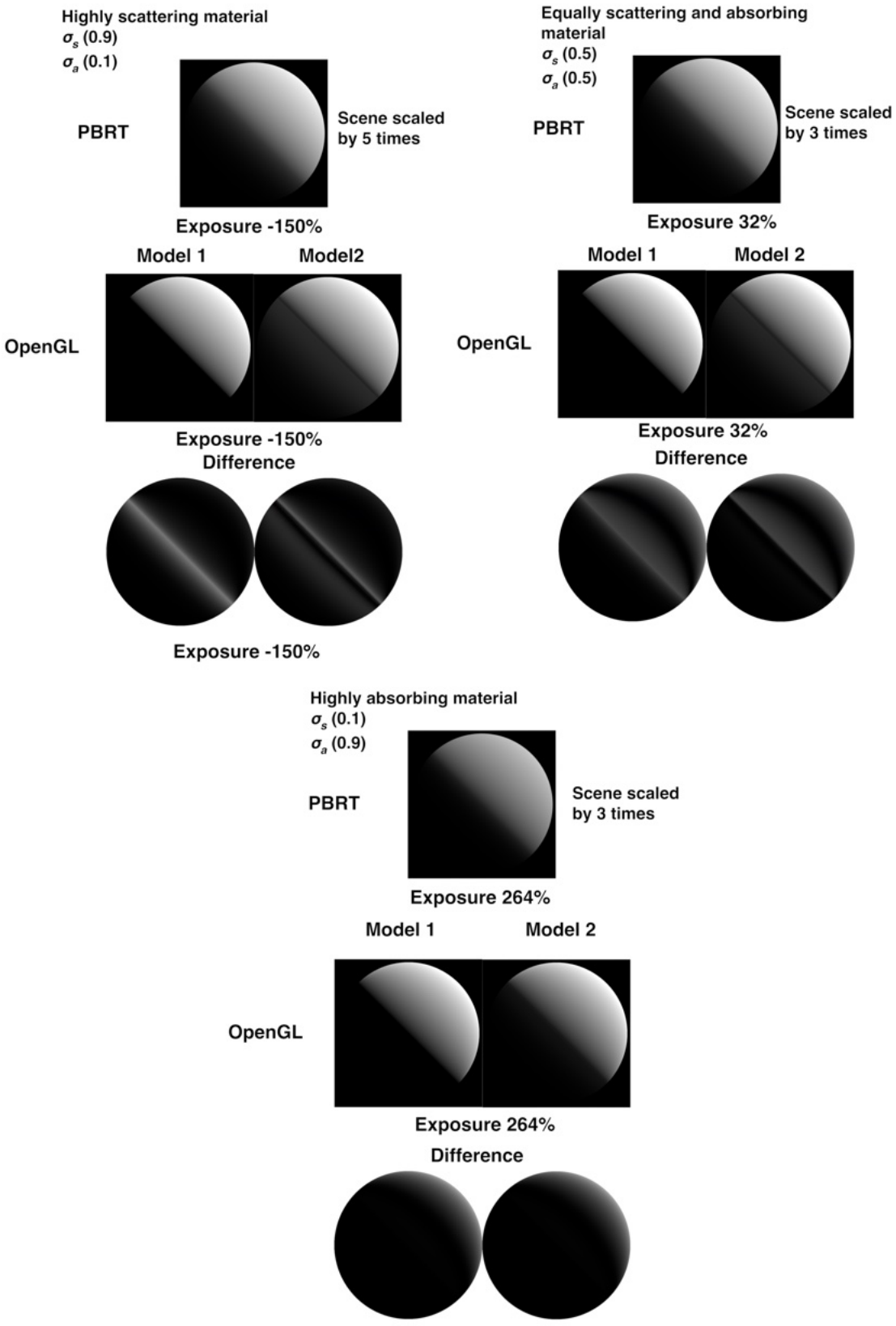


Figure 33: Rendered images of the object using different material properties in PBRT and OpenGL incorporating the second proposed model. Exposure settings for the rendered images are provided under each image.

8.7.4 Quantitative Comparison

As already mentioned in the visual comparison, the difference images confirm the quite favorable representation of light scattering within the object with increase in error values for the higher scattering material. The *Seams* between the two sides of the rendered objects are also pronounced in the comparison images and is the result of both single scattering and dipole method intersecting at the azimuthal line of incidence.

The next model described in Section 8.8 will attempt to address this issue by using multiple samples to evaluate the radiance of the object at the point of interest but for this model it was not expanded upon as the goal was to have a somewhat simple model to represent more translucent materials.

8.7.5 Error analysis

The mean RSE values are lower as a whole for this comparison, but that should be accredited towards the change in the setup of the scene as there are more fully dark areas in the images. The mean errors remain similar or are slightly lower using this model and show the biggest improvement for highly scattering material due to more pronounced translucency effect which is not portrayed at all from the first model.

The validation of the method comes from the difference images looking consistent across the backside of the object for lower scattering materials, indicating what material properties this model would be most suitable for.

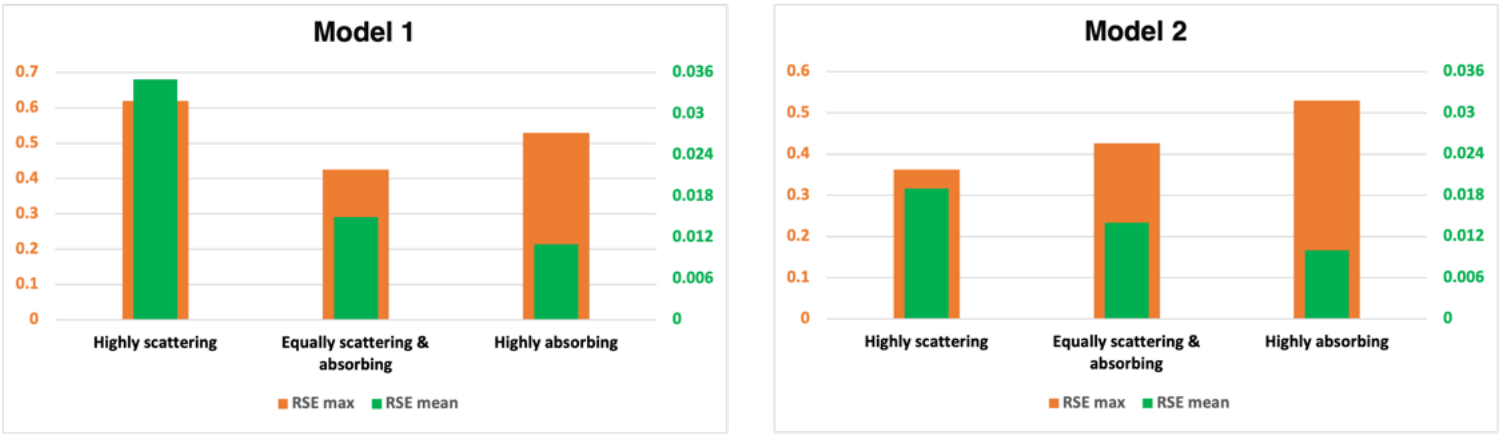


Figure 34: Graphs visualising RSE for 3 materials with different scattering to absorptions ratios for the first and the second subsurface scattering implementations.

8.8 OpenGL Model 3

The third model was developed to further improve on the accuracy of representing these more challenging translucent materials. In this section it will be explored how the model was improved upon using a different single scattering calculation and multiple sampling for the employed dipole method.

8.8.1 Single Scattering

To further improve the accuracy of subsurface scattering in OpenGL, the single scattering term was improved so that its contribution could also go past the fragments that are directly lit by the light source. This was done by leveraging the single scattering term described by Habel et al. [27], which is also used by PBRT (Section 8.1.9) also incorporating the distance between the points of ray entry and exit of the object in the calculation of the single scattering term for attenuation. This should allow for a more smooth transition between the two sides of the object and make the term generally more accurate for less than fully opaque materials.

A single scattering is much more pronounced within a short distance from the point of incidence for translucent materials [5], therefore evaluating radiance from a single sample point is not sufficient to accurately represent the term and requires sampling over an area.

8.8.2 Multiple sampling

To incorporate a revised version of single scattering term as well as to improve the accuracy of the diffusion approximation for multiple scattering, the dipole method was further improved by incorporating sampling over an area around the point of incidence. In this case the area taken into account was the full area of direct light incidence on the object to simplify the implementation since it is also expected for the granular material to be quite round and therefore it being a fitting approximation.

All the necessary data to account for multiple samples was already available at this point coming from the second model in the shape of stored texture data - the fragment coordinates and normal vectors from the perspective of the light source (Section 8.7.1). For this model, instead of finding an equivalent point on the front (from light's perspective) of the object, the whole area (the entire texture) was sampled to better approximate the light transportation through the medium to the point of interest - accumulating the dipole method calculations from each sample point. For a more accurate representation, especially with a more nuanced geometry, the area of sampling should be adjusted to be smaller and more focused, but considering the granular objects are quite round, this approach was sufficient and more time efficient way to validate the model.

Since the texture resolution was 1000×1000 pixels, sampling over the entire texture to evalute each point of interest (fragment) was not feasible in a single render pas at least for the machine used for the project, therefore a modifiable parameter was introduced to control the step size of the sampling. This is a quite naive approach as for every point of interest the same exact pattern of sample points are used, but as a proof of concept it was sufficient to show the improvement over a previous model and the benefits of sampling over an area.

Taking the area of sampling into account, the formula for the radiance of the object at the point of interest was adjusted to account for the multiple samples as follows:

$$L_o(x_o, \omega_o) = L_i(x_i, \omega_i) \frac{A}{N} \sum_{i=1}^{N} (S_d(x_i, \omega_i, x_o, \omega_o) + S^{(1)}(x_i, \omega_i, x_o, \omega_o)) \tag{60}$$

where:

- A - area of incidence
- N - number of samples
- S_d - multiple scattering component using the diffusion approximation from the second OpenGL model
- $S^{(1)}$ - single scattering term equivalent to one used in PBRT

8.8.3 Visual Comparison

To assess the improvement of over the last model, the exact same scenes were rendered as for the previous model consisting of a single light source at the top right of the object. The results are shown in Figure 36.

Initially striking difference is a visual artifacting on the incident area of the object, This issue stems from BSSRDF exponential decay with a distance from the point of incidence $e^{-\sigma_t d}$,where optically scaling the object increases the distances causing the scattering events to be more localized and requiring a more accurate sampling of the area to represent the object accurately [32]. Fortunately, granular materials are quite small therefore the effect should not be as pronounced when utilizing at a smaller scale.

8.8.4 Quantitative Comparison

While the visual results are not as spectacular as expected, the model does again show an improvement over the previous model especially for the high absorbing material

8.8.5 Error analysis

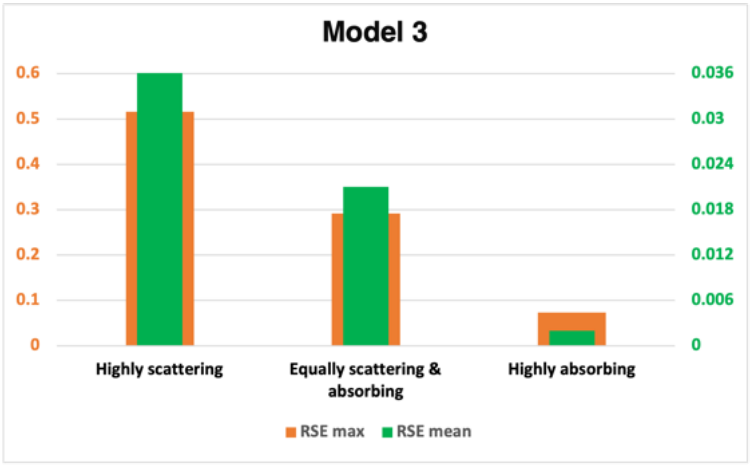


Figure 35: Graphs visualising RSE for 3 materials with different scattering to absorption ratios rendering a scene with a single light source.

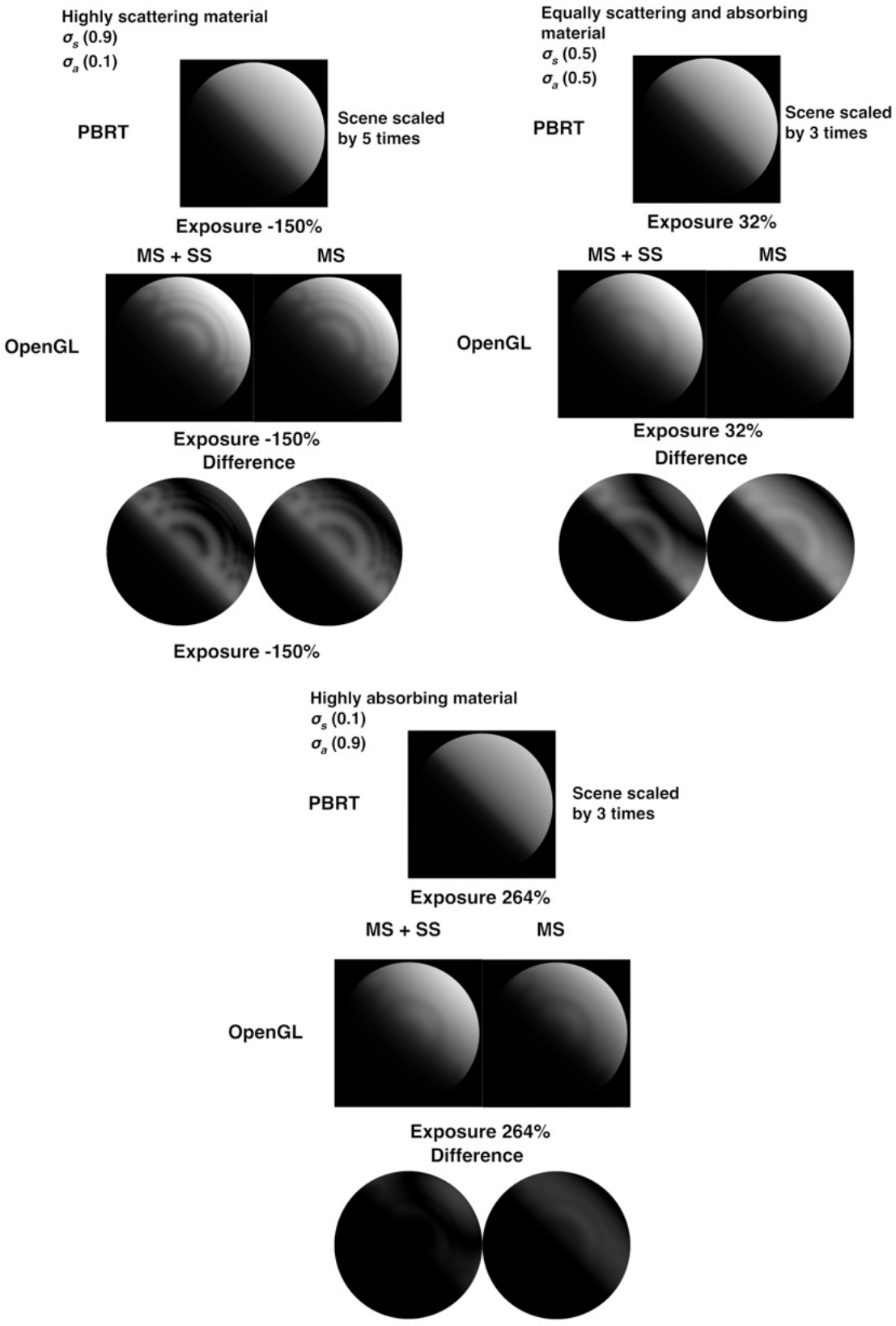


Figure 36: Rendered images form OpenGL model 3 implementation and PBRT and their difference images, showing RSE values. Separate images from OpenGL for multiscattering with single scattering (MS + SS) and just multiscattering (MS) Exposure of the images, if adjusted, labeled below each image.

8.9 Final Thoughts

Finally, to the last iteration of the OpenGL implementation of the BSSRDF model, the BRDF for specular from Section 7 is added to achieve the full BSSRDF model with the material describing parameters matching the ones used in PBRT. The images illustrating the example images can be seen in Figure 37. Due to errors and assumptions made analyzed in the previous sections, the results are not entirely matching the refference ones, but are nonetheless impressive considering time it takes to produce the images in PBRT compared to OpenGL. There are conundrums to solve in finding a suitable sampling strategy for the dipole method to have proper optical scaling of an object for the ability to represent objects in larger or smaller scales.

Three models for subsurface scattering were presented in this section, each improving upon the previous one. The first was a simple model approximating BSSRDF as a BRDF model for very optically thick materials, the second model incorporated the dipole method to simulate the light scattering within a semi-infinte medium, and the third model further improved upon the second model by incorporating multiple sampling to better approximate the light scattering within the object along with improvements to single scattering term calculation. In terms of rendering granular materials, each of these models could prove to be beneficial considering the material to be represented and the balance between visual results and computational complexity.

For example, looking at first model, it is most suitable for highly absorbing materials that tend to be rather opaque (as granular material is not likely to be opaque along with being high scattering due to its small size) as this will not impact the visual fidelity of the object and it is by far the most computationally efficient model that can easily be scaled up in real-time applications.

The second model, even given its visual limitations, could be suitable for materials that have some transparency to them, but are not fully transparent. Handling some transparency also allows for the implementation to be used when the material is lit from a light source on the opposite side of the camera since some light transport is approximated.

The last iteration of the implementation can be used with even more scattering objects making it a better fit to also represent more translucent granular materials

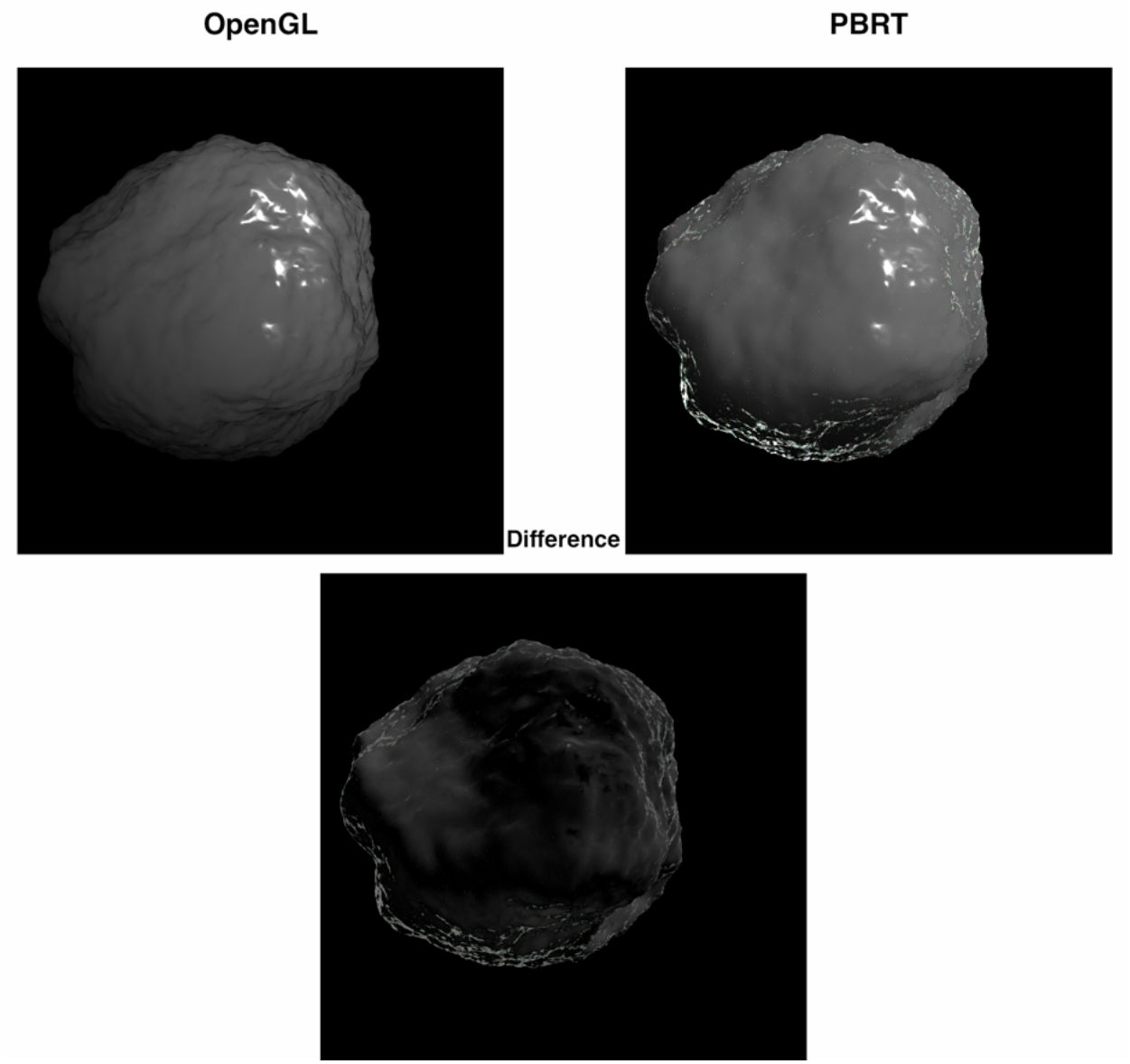


Figure 37: Rendered images of the object using the final OpenGL implementation of the BSSRDF model with the BRDF for specular. The material properties for image are $\sigma_a = 0.2$, $\sigma_s = 0.8$, $\eta = 1.6$ $roughness = 0.03$ $scale = 6$.

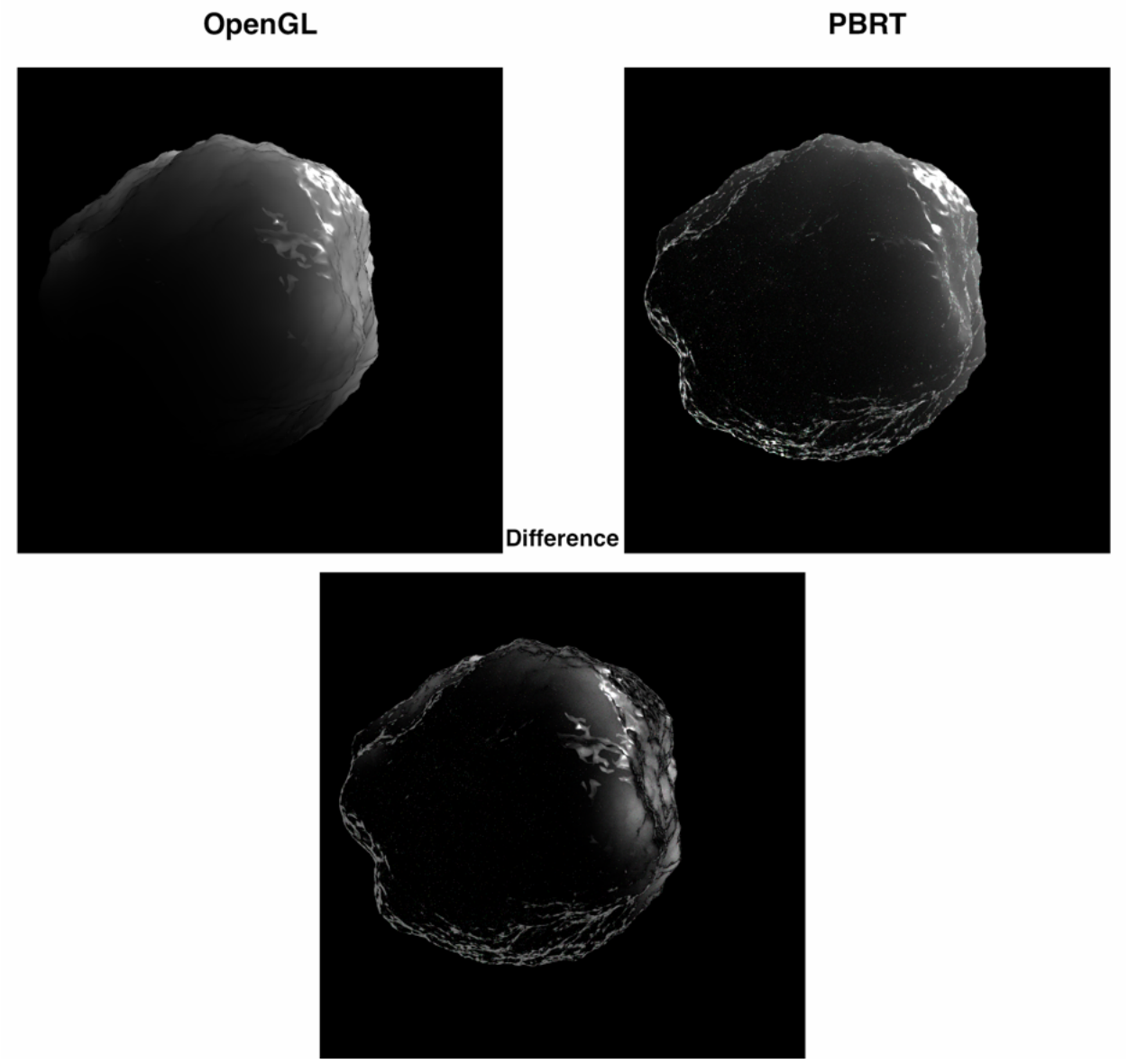


Figure 38: Rendered object with the same material properties as in Figure 37 but with the light placed in the opposite side to the camera.

9 Discusssion

For this project, the aim was to investigate the possibility of achieving a realistic rendering of granular materials using real-time rendering techniques and comparing them to more precise, but significantly more computationally intensive methods, in this case - rendering using Physically Based Rendering Toolkit. This section explores findings, limitations acknowledging inherent differences between these two rendering techniques, and potential future work to further expand on the results achieved here.

9.1 Overview of results

The look of a granular material is a complex phenomenon to simulate in real-time computer graphics, as it involves a variety of light interactions with the surface of the grain as well as within the volume of the grain. The results of this project show that the effects of more complex light interactions, such as subsurface scattering, can be approximated using real-time graphics APIs, such as OpenGL.

Performance wise, at this stage it is difficult to evaluate the performance of this method as it is expected to be used for a scene with numerous grains, However, there are possibilities for optimization, such as reducing the resolution of a texture used for depth information, making optimizations to to the shader code to use fewer instructions and more approximate calculations, precomputing non-fragment dependent calculations to reduce the number of calculations needed per fragment, and using instancing to reduce the memory usage of the scene.

9.2 Granular aggregate construction

Besides presented methods fitting closer with certain material properties, different models can be used to represent the grains as well. For instance, the upmost layer of the grain could be represented with a highly detailed approximation such as the *Model 3* described in this project and going deeper within the aggregate the grains could be represented with a less detailed models to reduce the computational cost of the rendering. At a certain depth within the aggregate the opposite side of the grain, would no longer be visible to the camera even at different angles, therefore it could be presented with significantly less detailed model - model 1.

9.3 Other viable techniques

Instead of approximating the BSSRDF of the grain in real-time, the lighting for a grain can be pre-computed using more precise tools and rendering engines. One such approach could be to use blender render baking to calculate the radiance data for each grain in the scene. This radiance data is equivalent to a result obtained from a rendered scene [33]. These pre-calculated results could then be used as a texture for a grain when rendering in real-time, reducing the computational cost of the procedure. However, this approach, while would certainly increase the realism of the rendering, would limit the ability to change the lighting conditions in real-time, as the radiance data would be pre-computed for a specific set of lighting conditions. Furthermore, if such solution is considered for a scene with large amount of objects (grains), there would be a need to have numerous sets of pre-computed data (assuming the grains are randomly rotated in the scene) resulting in a significant increase in memory usage therefore impacting the performance. Reducing the resolution of the pre-computed data could be a solution to this problem, but would result in loss of details in the rendering of the grains.

Other ways are to precompute diffuse normal maps for each separate color channel to simulate subsurface scattering for a specific object [34].

Non-object specific solutions incorporate precalculating light transfer for a set of vertices and storing the results as irradiance transport vectors that can be more efficienttly evaluated in real-time [35].

BSSRDF for multiple scattering used for for the implemented OpenGL method could also be precomputed into a texture and used as a lookup table for the shader, reducing the computational cost of the method since it is a function dependant only on the distance between the entry and exit points and the rest are just material describing properties.

9.4 Limitations

While uniform objects are simpler to render effectively using real-time rendering techniques, real-world objects are often more complex and in case of granular materials, the grains are often irregularly shaped, perhaps having some hairline cracks, creating glints of light, or being partially transparent - have inconsistent material properties throughout the volume of the grain. The proposed method does not account for these complexities, and does not include the effects of these inconsistencies on the appearance of the grain. Depending on the distance of the viewer from the object, these effects may not be noticeable, but for close-up shots, the realism of the rendering would affect realism of the scene.

The proposed methods, for the sake of simplicity and easier evaluation are presented on a spherical object, which is not representative of the real-world granular material. Using a more complex shape would require it to be substantially round or otherwise include a solution for self-shadowing, which, as is, is also not accounted for in the current state.

As the focus of this project was to investigate scenes with distant directional lights, additional work would be needed to extend the method to work with light sources that are not directionally uniform, requiring a more complex light transport framework. For example, accounting for the light scattering within the volume of the grain would require depth information from the perspective of the light source for each grain in the scene, which would be more computationally intensive. Additionally, the standard dipole approximation used in to simulate subsurface scattering effect, could be expanded to account for the direction of the incoming light in its calculation for enhanced precision as proposed by [7]

9.5 Other potential applications

While the focus of the project was on rendering granular materials the proposed methods are applicable to other materials. For larger scales, such as for rocks or pebbles, it could also be considered, but due to their increased size, the effects of subsurface scattering would be less pronounced, making the more simple approximations discussed for the first presented model more suitable in terms of computational cost and realism.

Hair and fur rendering could also benefit from the proposed method, as the subsurface scattering effects of light with hair are similar to those with grains of sand, with the hair strands acting as individual grains. But due to light scattering within the volume of the hair in a strongly anisotropic manner in the longitudinal direction [4], this would require further extension to account for such effect.

9.6 Future work

Brunt of the future work would focus on extending the proposed method for rendering granular materials to successfully incorporate numerous grains in a scene, where these grains would be capable of interacting with each other. More specifically, the method would need to account for the grains occluding each other affecting the lighting conditions on the grains that are occluded for realistic perceived depth of the object made out of grains. This phenomenon is particularly important for light colored granular materials, where forward scattering is more pronounced, and the illumination penetrates deeper into the volume of the object made out of grains. Such inclusion would extend the rendering method to have dual scattering, where to an existing approximation to local scattering for each individual grain in the scene an addition of global multiple scattering would be made, as described for hair rendering in [4].

A realistic approach for global scattering could be implemented as described by Dal Corso et al. [32], where the author proposes evaluating scattered radiosity on an object by sampling transmitted irradiance for numerous points across is surface. Virtual light sources are then positioned over the object's surface, used to distribute the evaluated radiosity. These virtual light sources are then used to simulate indirect illumination throughout the scene and could be used to simulate the global scattering of light within the volume of the object made out of grains.

Furthermore, To have an enormous amount of grains in the scene - an object made of individual grains, it is not feasible to have vertex data for each single grain, as it would result in a significant use of memory to store the vertex data for each grain in the scene. Instead, instancing should be incorporated, where a single grain model would be used to render all the grains in the scene, with each grain only having a unique transformation matrix. This would significantly reduce the memory usage, as only additional transformation matrices would need to be stored for each grain in the scene, making more complex scenes less memory intensive to render in real-time [36].

After other potential improvements to subsurface scattering approximation for a single grain are made, evaluating against PBRT's capabilites of rendering an object as a dielectric surface with a medium inside using a radiative transfer equation, would be a good way to further evaluate the level of realism achieved.

10 Conclusion

To conclude the presented work, the aim of this project was to investigate the rendering of granular materials at a mesoscopic scale - rendering individual grains and accounting for their look as an aggregate instead of approximating the material as a whole.

A way was established to quantify the difference besides visual comparison between the proposed methods and a physically accurate offline renderer - PBRT. Diffusion comparison was used to establish the baseline of error for the comparison of the renderers and to establish exactly matching conditions for comparisons performed afterwards.

A specular reflection model was implemented in OpenGL to simulate the reflection of

light off the surface of a grain in a physically accurate manner with the results verified against PBRT.

3 potential methods to represent a granular material were proposed leveraging different complexities of the BSSRDF model to approximate the subsurface scattering effect of light within the volume of the grain. These 3 methods were further analysed for their potential use cases and accuracy.

In the discussion section, future of this work was discussed to address the limitations of the proposed methods and to further extend the method to work with numerous grains in a scene, where the grains would be capable of interacting with each other.

For a master student in computer science, this was an invaluable experience where I got to experience using offline renderer PBRT and obtained a deeper understanding of the process behind ray tracing and path tracing. I also got to experience the process of implementing more complex scenarios in a reak-time renderer, further expanding my knowledge of computer graphics and rendering techniques. Based on that alone I would consider this project a success!

A Appendix A

The source code related to the project is available on Github : `https://github.com/nnaglis/MSC_thesis.git`

References

[1] T. Müller, M. Papas, M. Gross, W. Jarosz, and J. Novák, "Efficient rendering of heterogeneous polydisperse granular media," *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, vol. 35, pp. 168:1–168:14, Dec. 2016.

[2] J. Meng, M. Papas, R. Habel, C. Dachsbacher, S. Marschner, M. Gross, and W. Jarosz, "Multi-scale modeling and rendering of granular materials," *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, vol. 34, July 2015.

[3] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 4 ed., 2023.

[4] A. Zinke, C. Yuksel, A. Weber, and J. Keyser, "Dual scattering approximation for fast multiple scattering in hair," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, vol. 27, no. 3, pp. 32:1–32:10, 2008.

[5] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan, "A practical model for subsurface light transport," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*.

[6] A. Dal Corso, "Real-time rendering of translucent material with directional subsurface scattering," Master's thesis, University of Padua, 2015.

[7] J. R. Frisvad, T. Hachisuka, and T. K. Kjeldsen, "Directional dipole for subsurface scattering in translucent materials," tech. rep., Technical University of Denmark, Department of Applied Mathematics and Computer Science, 2014.

[8] "GLFW Documentation." https://www.glfw.org/docs/latest/.

[9] LuTheCoder, "How to setup glfw on new macs | glfw." https://youtu.be/6AHq0jTrypw?si=xEN7jT8zBopX8coE.

[10] "GLAD Source code." https://github.com/Dav1dde/glad.git.

[11] M. Pharr, W. Jakob, and G. Humphreys, "Pbrt file format v4." https://www.pbrt.org/fileformat-v4, 2023.

[12] "Open Asset Import Library." https://github.com/assimp/assimp.

[13] J. de Vries, "Learnopengl codebase." https://github.com/JoeyDeVries/LearnOpenGL, 2024.

[14] J. B. Nielsen, H. W. Jensen, and R. Ramamoorthi, "On optimal, minimal brdf sampling for reflectance acquisition," vol. 34, no. 6, 2015.

[15] Y. Salih, A. S. Malik, N. Saad, *et al.*, "Tone mapping of hdr images: A review," in *2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012)*, vol. 1, pp. 368–373, IEEE, 2012.

[16] "Pixelmator Pro." https://www.pixelmator.com/pro/.

[17] "OpenGL Reference Pages." https://registry.khronos.org/OpenGL-Refpages/.

[18] S. L. Bangare, A. A., S. Mishra, and M. Kasar, "Reviewing otsu's method for image thresholding," *ResearchGate*, 2015. Available at: https://www.researchgate.net/profile/Sunil-Bangare-2/publication/282282124_Reviewing_Otsu's_Method_For_Image_Thresholding/links/57c6e3e408ae9d64047e90dc/Reviewing-Otsus-Method-For-Image-Thresholding.pdf.

[19] R. L. Cook and K. E. Torrance, "A reflectance model for computer graphics," ACM, 1982.

[20] B. Walter, S. Marschner, H. Li, and K. E. Torrance, "Microfacet models for refraction through rough surfaces," Eurographics Association, 2007.

[21] M. Pharr, W. Jakob, and G. Humphreys, "Pbrt users guide." https://pbrt.org/users-guide-v4.

[22] "Pbrt-v4: Source code." https://github.com/mmp/pbrt-v4.

[23] S. Green, "Real-time approximations to subsurface scattering," in *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics* (R. Fernando, ed.), ch. 16, pp. 263–278, Addison-Wesley, 2004.

[24] J. Meng, M. Papas, R. Habel, C. Dachsbacher, S. Marschner, M. Gross, and W. Jarosz, "Multi-scale modeling and rendering of granular materials," in *Proceedings of the 42nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2015)*, ACM, 2015.

[25] J. Stam, "Multiple scattering as a diffusion process," in *Eurographics Rendering Workshop 1995*, Eurographics, June 1995.

[26] H. Weber, "The fresnel equations for lossy dielectrics and conservation of energy," *Journal of Modern Optics*, vol. 61, no. 15, pp. 1219–1224, 2014.

[27] R. Habel, P. H. Christensen, and W. Jarosz, "Photon beam diffusion: A hybrid monte carlo method for subsurface scattering," in *Eurographics Symposium on Rendering* (N. Holzschuch and S. Rusinkiewicz, eds.), vol. 32, Eurographics Association, 2013.

[28] C. C. Grosjean, "A high accuracy approximation for solving multiple scattering problems in infinite homogeneous media," *Il Nuovo Cimento (1955-1965)*, vol. 3, pp. 1262–1275, June 1956.

[29] E. D'Eon and G. Irving, "A quantized-diffusion model for rendering translucent materials," *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011)*, vol. 30, July 2011.

[30] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 3rd ed., 2018.

[31] J. de Vries, "Learnopengl." `https://learnopengl.com/`, 2024.

[32] A. Dal Corso, J. R. Frisvad, J. Mosegaard, and J. A. Bærentzen, "Interactive directional subsurface scattering and transport of emergent light," *The Visual Computer*, vol. 33, pp. 371–383, Mar. 2017.

[33] "Blender 4.1 manual, render baking." `https://docs.blender.org/manual/en/latest/render/cycles/baking.html`.

[34] W. Ma, A. Jones, T. Malzbender, T. Hawkins, and H. Fuchs, "Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[35] R. Wang, J. Tran, and D. Luebke, "All-frequency interactive relighting of translucent objects with single and multiple scattering," *ACM Transactions on Graphics*, vol. 24, pp. 1208–1217, July 2005.

[36] P. Shirley and S. Marschner, *Fundamentals of Computer Graphics, Fourth Edition*. CRC Press, 2015.