



DTU Compute
Department of Applied Mathematics and Computer Science
January, 2022

Real-Time Rendering of Granular Materials

Master Thesis

Author: Nynne Kajs (s193156)

Supervised by: Jeppe Revall Frisvad and Jakob Andreas Bærentzen



Real-Time Rendering of Granular Materials

Danish Title: Rendering af Granulære Materialer i Realtid

Master Thesis

January, 2022

By

Nynne Kajs

Supervised By

Jeppe Revall Frisvad, Associate Professor, Department of Applied Mathematics and Computer Science, Technical University of Denmark

Jakob Andreas Bærentzen, Associate Professor, Department of Applied Mathematics and Computer Science, Technical University of Denmark

Copyright: Reproduction of this publication in whole or in part must include the customary bibliographic citation, including author attribution, report title, etc.

Cover photo: Nynne Kajs, 2021

Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby Denmark

www.compute.dtu.dk

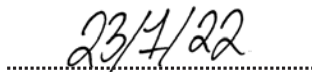
This thesis has been prepared over five months from August 23, 2021 to January 23, 2022 at DTU Compute, Department of Applied Mathematics and Computer Science, at the Technical University of Denmark, for 30 ECTS as part of the degree Master of Science in Human-Centered Artificial Intelligence, MSc Eng. The thesis has been prepared in collaboration with the game company Playdead.

It is assumed that the reader has knowledge within the areas of computer graphics and rendering.

Nynne Kajs - s193156

A handwritten signature in black ink, appearing to read 'Nynne Kajs', written over a horizontal dotted line.

Signature

A handwritten date '23/4/22' in black ink, written over a horizontal dotted line.

Date

Abstract

This project is done in collaboration with the game company Playdead and investigates to which degree a real-time solution can be used to approximate the appearance of real granular materials, while addressing the challenges of rendering granular materials in real-time. Currently, no real-time solution effectively deals with rendering of granular materials at multiple scales while accounting for all the shading contributions necessary for accurately representing the appearance of the materials.

This project focuses on the granular material sand and the methods of this project are derived from observations of photos and videos of real sand as well as realistic offline pathtracing results. This project handles shading at close distances differently than shading at greater distances, representing the granular surface by its micro structure at close distances and transitioning to a BRDF representation for increasing distances. The micro surface structure is represented by a grid arrangement, where each grid cell is represented by cube normals rotated using pseudo random noise seeded by the object position of the fragment. It models diffuse, specular, and transmissive contributions that are regulated by user-controlled parameters for colors, subsurface scattering, specular roughness, transmission, transmission roughness, and porosity.

This project, compared to other current real-time approaches, considered the connection between offline shader parameters and their influence on the appearance of sand and attempted to recreate these observations in real-time. The methods of this project were able to produce real-time results that qualitatively approximated expensive offline methods and to a degree the appearance of real sand at multiple scales, accounting for relevant shading contributions.

Keywords — Granular materials, real-time rendering, rendering of sand, multi-scale materials

Foreword

I'd like to thank my supervisors Jeppe Revall Frisvad and Jakob Andreas Bærentzen for sharing their knowledge and their time during this project. I am immensely grateful for the level of enthusiasm to which they have approached guiding me with this project, something that has been a great motivational factor for me throughout the entire process.

To the people of Playdead, I am forever grateful for the amazing opportunity I have been given of working with you during this project. A special thanks to my inspiring supervisors at Playdead, Mikkel Gjøl and Jakob Gath, who took their time to share their expertise and help along the way. Thanks to everyone from Playdead for being incredibly welcoming and friendly, the time I have spent in your office is a time I will value for the rest of my life.

To my friends, thank you for making sure I took the time to relax when you could tell I needed it. The fun times we shared along the way ensured I maintained motivation, ultimately improving the outcome of this project.

Finally, I want to thank my parents for letting me stay at their place and making sure I was well fed and caffeinated during the last intensive weeks of working on this project.

Contents

Preface	ii
Abstract	iii
Foreword	iv
1 Introduction	1
1.1 Scope	2
1.2 Problem Statement	3
2 Analysis	4
2.1 Appearance Study	4
2.2 Pathtracing Sand	9
2.3 Analysis Conclusion	21
3 Litterature Review	24
3.1 Multi-Scale Modeling and Rendering of Granular Materials	24
3.2 An Analytic BRDF for Materials with Spherical Lambertian Scatterers	25
3.3 Real-Time Rendering of Procedural Multiscale Materials	26
3.4 Real-Time Sand Rendering in Journey	28
4 Methods	31
4.1 Micro Normals	31
4.2 Shading	36
4.3 Transitioning from Micro to Macro Scale Shading	52
4.4 Method Conclusion	58
5 Implementation	60
6 Results	61
6.1 Parameter Comparison	61
6.2 Pathtracing Comparison for HDRI Matching	67
6.3 Breakdown of Shading Contributions	72
6.4 Test Builds	74
6.5 Company Feedback	75
7 Discussion	77
8 Conclusion	80
9 Future Work	81

Bibliography	83
A Appendix 1 Shader Implementation	86

1 Introduction

Rendering granular materials accurately at multiple scales in real-time remains a challenge. Granular materials in computer graphics represent the real-life occurrences of individual granules, such as sugar, salt, snow, and sand grains, acting together as a collected unit. These materials appear wildly different depending on their distance to the observer. For instance, we can pick up a handful of sand and carefully inspect each individual sand grain, as in Figure 1.1a, or we can overlook a beach where individual sand grains are no longer discernible but together make up uniformly appearing sand dunes, depicted in Figure 1.1.



(a) Handful of sand grains [1].

(b) Sand at a beach [2].

Figure 1.1: Handful of sand grains and sand at a beach. These images show how vastly different granular materials appear depending on their distance to the observer, and reveal just one of the many challenges related to accurately representing and rendering the materials in real-time.

Representing this multi-scale phenomenon is not a trivial computer graphics problem. When rendering a granular material, the individual granules can take up several pixels in the image, or each pixel can cover thousands of granules, both cases often occurring in the same rendered image. Therefore, not only representations of granular materials at multiple scales should be handled, a proper transition between the scales must be managed. Adding to the complexity is the fact that granules rarely appear strictly diffuse, specular, or transmissive, but rather interact with light in a multitude of ways, which directly affects the appearance of the granular material at macro scale. Meng et al. [3] proposed an offline rendering approach that effectively deals with the aforementioned challenges. They utilize complex geometric representations of granules with both reflective and refractive material properties and use pathtracing to accurately trace light paths through the granules of the aggregate objects. They present highly realistic results for granular materials at multiple scales and effectively handle the transition from micro to macro representation. Unfortunately for real-time solutions, the possibilities are significantly more re-

stricted. Having realistic representations of each granule and accurately tracing light paths through the collection of granules is not feasible. Naturally, approximations must be made, however the process of determining optimal approximations is not straight-forward. Deon et al. [4] proposed a BRDF model for representing porous or granular materials. While BRDF methods can be good approximations for granular materials at macro scale, they are not suitable for representing the granularity at micro scale, and often do not account for all relevant material properties. To the authors knowledge, no current real-time solution convincingly approximate the appearance of granular materials at multiple scales, accounting for the transition between scales and all relevant shading properties. This project investigates how a real-time solution can be used to deal with the presented challenges for rendering granular materials, and to what extent the necessary real-time approximations are representative of real-life observations.

1.1 Scope

This project is done in collaboration with the game company *Playdead*. Playdead needs a granular material rendering approach that is suitable for real-time applications, specifically for games. The approach should include representations of granular materials at multiple scales, as it will be observed by the player from multiple distances in the game. Furthermore, the company desires an approach that represents the granular material as a surface while enabling the possibility of single grains detaching from the surface and being represented in particle form. Playdead has experimented with using the micro facet reflectance model Oren-Nayar [5], and while the results of using this model were sufficient for some cases, it falls short in others. Specifically, it does not correctly model light arriving at the observer at angles where the light source is on opposing sides of the surface in relation to the camera. Furthermore, Oren-Nayar does not correctly model the granularity of the material at a close scale and does not take into consideration that granules are rarely completely diffuse. Finally, as the solution of this project is created to be used in a creative process in a game company, it should allow for artist directability, i.e. exposing relevant parameters in a meaningful way to allow for creative expression to suit the company's vision.

These requirements have been summarized below.

- **Real-time.** The implementation should allow for the company to render granular material meshes that take up most of the space on the rendered images in real-time.
- **Multiple scales.** The granular material should be represented at multiple scales, as it will be observed at multiple distances.
- **Surface to particle representation.** The granular material should be represented as a surface while allowing singular grains to detach from the surface and be represented as particles.
- **Solve issues with existing microfacet models.** The implementation of this project should consider the issues the company has faced with existing models and offer a solu-

tion to these.

- **Artist directability.** The solution should allow for artist directability to ensure it can act as part of a creative workflow where artists can perform tweaks to match the artistic expression that is desired.

The granular materials that Playdead is specifically interested in rendering at the current time is sand and snow. It was decided for this project to focus specifically on sand as a means to narrow the scope of the project. All granular materials share similar properties and similar problems must be tackled for an implementation of each, however their differences are too great to allow for a completely generalized solution to be made within the scope of this project. It is however believed that while focusing on sand, the findings can be generalised to other granular materials and used as a foundation for further research.

1.2 Problem Statement

This led to the following problem statement. As described, as a means of narrowing the scope of the project, sand was chosen as a focus which explains the formulation of the problem statement. However, the overall theme of this project is rendering of granular materials, and it is believed that the findings for sand can act as a stepping stone for investigating how a real-time approach can be used to render granular materials in general.

How can a real-time rendering approach be used to approximate the appearance of sand at multiple scales, accounting for its relevant material properties?

2 Analysis

This project deals with creating a real-time solution that approximates the appearance of sand, dealing with the challenges presented in the introduction. This section carries out an analysis in which photographs of sand will be studied to get an estimate of the properties of real sand at multiple scales. After this, a field trip will be conducted to a beach at which videos will be shot. This will guide the aspects of the real-time solution that cannot be experienced using photos alone. In terms of shading, it can be difficult to understand the link between the material properties of individual granules and the appearance of the granular surface at macro level. Therefore, this project will utilize an offline rendering approach in which a collection of sand grains can be realistically modelled and pathtracing be used to get accurate rendering results. Hereby, material properties of individual granules can be adjusted and how this affects the granular material at macro level can be inspected. Having a physically accurate offline rendering approach provides a frame of reference for a real-time solution and allows for investigating to what extent a real-time solution can model the observations made for offline pathtracing, and where it falls short.

2.1 Appearance Study

2.1.1 Photos of Sand

In this section, photographs of sand at multiple scales will be presented and analysed. In Figure 2.1, closeups of nine different types of sand can be seen.

The types of sand depicted in this picture are far from exhaustive to describe the types of sand that exists, however they do provide the understanding that the properties of sand vary greatly. The properties that most noticeably differ within each type of sand are color, shape, size, transmissiveness and specularity. Looking at the first type of sand (1), it can be seen that the grains vary greatly within each category. The colors of the grains include black, white, green, red, and even more. Some are relatively transmissive, others diffuse or specular. They differ distinctly in size as well as shape. Overall, these properties result in a type of sand that have distinctively varied types of grains within each category. Looking to sand type (2), a great variance can also be observed in color, diffuseness, specularity, and transmissiveness, however the overall shape and size of the grains are very similar, resulting in a very different appearance compared to sand type (1). Some sand grains for sand type (5) have a milky appearance, suggesting subsurface scattering. This is mostly apparent for the elongated sand grain in the upper left corner. Looking across all types, the distributions of differences within each category can be observed to be responsible for the distinctive appearance of the specific types of sands.

Figures 2.2 and 2.3 depicts pictures of the same sand from Kelso Dunes, California at increasingly greater distance from the observer.



Figure 2.1: Close up of different types of sand [6]. From top row to bottom row, left to right; (1) glass sand (Kauai, Hawaii), (2) dune sand (Gobi Desert, Mongolia), (3) quartz sand with green glauconite (Estonia), (4) volcanic sand with reddish weathered basalt (Maui, Hawaii), (5) biogenic coral sand (Molokai, Hawaii), (6) coral pink sand dunes (Utah), (7) volcanic glass sand (California), (8) garnet sand (Emerald Creek, Idaho), (9) olivine sand (Papakolea, Hawaii).



Figure 2.2: Closeup of sand and bug from the Kelso Dunes (California) [7]

Looking at the same type of sand from multiple distances allow for seeing the specific changes happening for increasing distances without the appearance being influenced by varying types of sand grains. In the first image (Figure 2.2), a closeup of the sand at Kelso Dunes, California, can be seen. As for the images in Figure 2.1, a distinct distribution of different types of sand grains can be observed. A dominant beige color is present, while several less dominant colors can

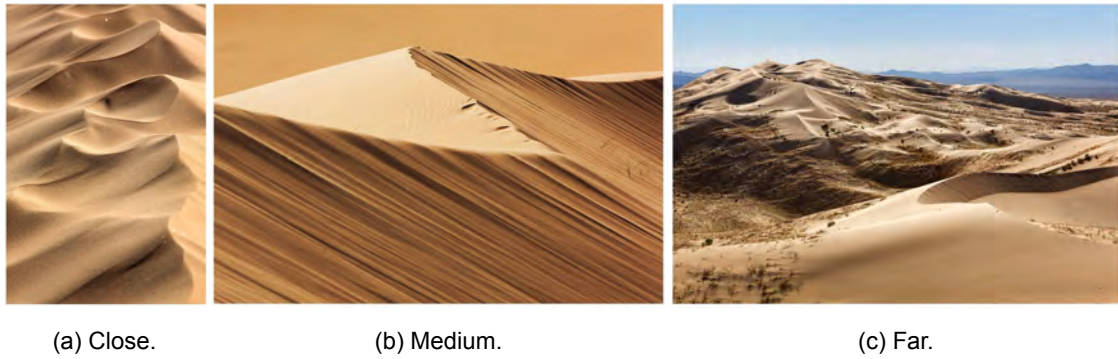


Figure 2.3: Pictures from the Kelso Dunes (California) at different distances [8].

be observed, such as black, red and white. Furthermore, the grains differ in how transmissive they are, ranging from appearing close to glass to completely opaque and stone-like. At the farther distance depicted in Figure 2.3a, a noticeable effect comes from the shadows falling on individual grains due to how the grains are stacked across the surface in relation to each other. Some grains lie perfectly illuminated while others are occluded by other grains. This gives a varying granular appearance across the surface. The characteristics of individual grains observed in Figure 2.1 and Figure 2.2 are no longer apparent. This can be explained by the fact that, compared to the other figures, the sand is now depicted at a distance where each pixel covers more sand grains. Another dominant appearance expression comes from the shape of the dunes, creating highlighted areas and areas in shadow. Furthermore, very noticeably are the small specular highlights, so-called *glints*, in which concentrated light is reflected off a sand grain to the observer. Figure 2.3b provides a good visualization of the transition across scales. In the foreground, the observations made for Figure 2.3a can still be seen, however in the background of the image at greater distances, each pixel covers more and more sand grains resulting in a more diffuse appearance for the sand. At this distance, glints can no longer be observed, and likewise the graininess arising from individual grains either being either illuminated or occluded by neighbouring grains is no longer apparent. The observations made for the background of Figure 2.3b can more easily be inspected in Figure 2.3c. At this distance, the overall shape of the dunes is what contributes mostly to the appearance of the dunes, and the influence of the micro surface structure can no longer be discerned in the image. This image further demonstrates the diffuse appearance of the sand at macro scale, where each pixel covers thousands of sand grains. Another very noticeable effect is the highlighted areas at glancing angles, especially apparent at the top of the dunes.

The appearance observed especially at macro scale for greater distances arguably depend considerably on the position of the observer and the light. For instance, at macro level, the noticeable highlighted dune peaks appear to be a result of more light from the environment being reflected towards the observer at glancing angles. Therefore, observing the dunes from another angle most likely yield a very different appearance. The shadows arising from the macro surface shape also depend on the direction of light. At closer distances, glints can

be observed. As glints arise from light being reflected off small areas on individual grains to the observer, they appear at each sand grain only at very specific angles. Therefore, this observation likewise depend on the position of the observer and light source. This means that other observations could potentially have been made if images from other times of day, other observer positions etc. had been analysed. This could have provided other perspectives on the appearance of sand, something that is potentially overlooked using a small, finite set of images. Furthermore, analysing increasing distances for other types of sand than the sand from Kelso Dunes could provide different insights, or looking at beaches instead of dunes etc. The following section will examine some of the effects that occur for varying observer positions as well as investigate smooth transitions from micro to macro scale observations.

2.1.2 Field Trip

A field trip was made around 9-10 AM in December to Bellevue Beach in Klampenborg, Denmark to study sand in real life. Videos were made using an iPhone X to capture the change in appearance of sand when transitioning smoothly from observing the sand at different distances, as well as the appearance of glints when as the observer position changes. The videos have been attached in the Appendices of this project, however they can also be accessed using the links in this section.

The first video (accessible [here](#)) shows the transitioning from filming sand at a great distance to filming closeups of sand. Auto focus was used to allow the relevant parts of sand to be in focus. Figure 2.4 shows selected views from the video, namely one where the sand is observed closely and one from a greater distance. The video shows how the appearance of sand changes given the distance to the observer. It essentially shows the same as was seen in the appearance study, however explicitly shows the smooth transition from close scale to far. At a distance, the macro structure of the sand dominates the appearance, especially towards the left of the frames where the ripples of sand combined with the rising sun creates strong shadows. In the right side of the video towards the fence however, a path of flat sand can be seen. As for sand at distances presented in the previous section, this patch has a diffuse appearance. The video shows that the observer actually has to be quite close to discern the shape and color of individual grains, however the glints and shadows of individual grains have a strong influence on the overall appearance at a considerable distance to the observer. This video can aid the understanding of how an implementation should model the smooth transition from the more uniform appearance at a greater distance to granular appearance at close scale.

The second video (accessible [here](#)) was made to capture the glints in the sand. In real life, the glints were a very noticeable part of the appearance of sand when observed closely and still for a considerable viewing distance while standing. However, despite being very noticeable in real life, they were difficult to capture on camera. It was therefore ensured that the glints remained unfocused in the video, as this enhanced them, and the effects of changing observer position could be observed. The video shows the distinct effect that glints have as the observer position varies; the small specs of light appear and disappear at different speeds, depending on



(a) Close.

(b) Far.

Figure 2.4: Pictures taken at Bellevue beach in Klampenborg, Denmark. **Left:** closeup of sand. **Right:** sand at a distance.

how quickly the observer moves, how far away they are from the observer, and the size of the facets they are reflected from. The latter also affects how large the glint becomes in the image. Furthermore, the glints seem approximately uniformly distributed across the sand. The color of the glints appear to match the color of the sun, and the intensity of the glints is considerably higher than the reflected light of the neighbouring sand grains. Figure 2.5 show two still frames from the video. It can be seen that between the two frames, some glints gain strength (yellow dotted box), some remain the same (blue dotted and striped box) and some lose strength (red solid box).

These videos will be used to guide the real-time implementation of this project, namely how the transition from observing the sand closely and from a greater distance should be made smoothly and how glints change as the observer moves. As described in the previous section, dominant appearance properties of the sand depend on the lighting conditions. More videos could have been shot during different times of day or in a controlled setting, but this was deemed infeasible during the project. Furthermore, the frequent small ripples in the sand combined with the rising sun dominated the appearance of the sand and made for a challenge for estimating the properties of sand related specifically to shading, and not just the macro shape. However, the knowledge gathered in this section still gives a good insight into some of the interesting real-time effects noticeable on sand, such as glints.



(a) Frame 14.

(b) Frame 16.

Figure 2.5: Pictures taken at Bellevue beach in Klampenborg, Denmark. **Left:** still image from video of glints at frame 14. **Right:** still image from video of glints at frame 16. Three points have been highlighted in both images; a glint that gets stronger (yellow dotted box), a glint that remains virtually of same strength (blue striped and dotted box), and glints that get weaker (red solid box) from frame 14 to 16.

2.2 Pathtracing Sand

As described, this project utilizes an offline rendering approach as a frame of reference for a real-time implementation. Compared to exclusively studying the appearance of real sand, digitally representing sand and using pathtracing allows for acquiring physically accurate results that can be made under controlled conditions and can be compared to a real-time solution. This can be used to measure how well the real-time rendering approach of this project approximates realistic results. It furthermore allows for exploring how the material properties of individual sand grains affect the appearance of the surface of sand at macro scale. *Blender* [9] and its physically-based pathtracer *Cycles* was used for the purpose.

Individual sand grains were modelled inspired by the closeup of sand grains in Figure 2.1. Only one type of sand grain geometry was modelled and used for all sand grains. The modelled sand grain can be seen in Figure 2.6. The figure is merely intended to show the geometry of the modelled sand grain, and the final material properties of the grains for the pathtracing results will be explained later in this section.

An HDRI of a beach (see Figure 2.7) was chosen to light the scene.

One of the purposes of using pathtracing was to investigate how changing material properties affected the appearance of individual sand grains and the collection of sand grains at macro scale. Therefore, renders were done of individual sand grains, sand grains scattered across a sphere and finally sand grains scattered across a *wavy plane*, i.e. a plane with a modifier that added a single wave in an upwards direction. The wavy plane was intended to represent a dune-like shape. The sphere and plane were layered, meaning the sphere was constructed of several spheres of decreasing size positioned inside each other, whereas the wavy plane was constructed of several wavy planes of the same size stacked on top of each other. The number

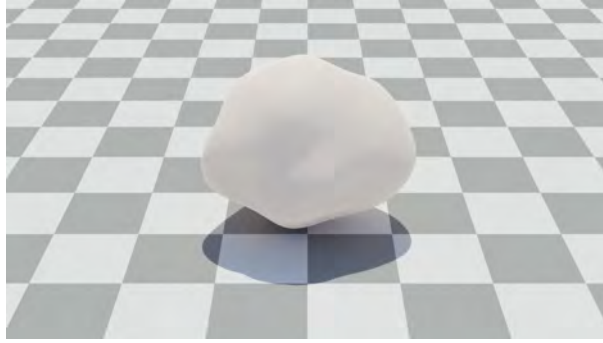


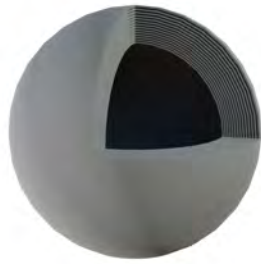
Figure 2.6: Geometry of the modelled sand grain. Note, that the material properties are arbitrary and the used material properties are explained later in this section.



Figure 2.7: HDRI [10] used for lighting the scene used to render sand in Blender.

of layers were chosen to be the minimum of what ensured no light would travel from the inside of the sphere to the outside and vice versa, and from the top of the wavy plane to the bottom of the wavy plane and vice versa. This was ensured by having a green *debug sphere* and a green *debug plane* at the inside of the sphere and the bottom of the plane respectively, and checking if any green light was visible in the render of the sphere and plane. In total, 14 spheres and 14 planes were used. The geometry of the sphere and wavy planes can be seen in Figure 2.8.

For the sphere and the wavy plane, sand grains were distributed across each layer using Blenders *Geometry Nodes* using the *Poisson Disc* distribution option, and each sand grain was assigned a random rotation. For each layer in the sphere and wavy plane, a different seed was chosen for the Poisson disc distribution to make each layer different. The camera was positioned on the opposite side of the sand objects in relation to the main light source (the sun) in the HDRI at a height that allowed for the main light source to be reflected to the camera from



(a) layered sphere



(b) Layered wavy planes

Figure 2.8: Geometry for objects used for pathtracing. The sphere consist of 14 layers of decreasing sized spheres, and the wavy plane of 14 layers of planes with a curve modifier. For visualization purposes, the sphere is depicted with a hole that allows to see the whole geometry.

the top of the sand objects (see Figure 2.9). This was decided to allow for as many effects as possible to be observed from the sand grains, i.e. transmission of light, reflection of light etc.

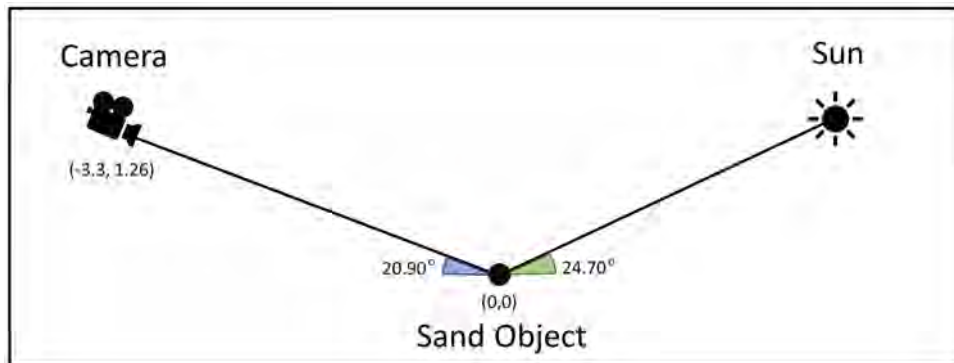


Figure 2.9: Setup for sand renders in Blender. The *Sand Object* refers to the sand grain, sphere, and wavy planes. The sun represents the approximate direction towards the the sun in the HDRI, and therefore not a position. The camera, approximate sun direction, and sand object all share the same y-coordinate in Blender, and the setup has therefore been depicted in 2D, merely showing the elevation angle of the camera and sun.

Blender’s *Principled BSDF* shader was used for the material of the sand grains. This shader is based on Disney’s PBR shader [11]. It has a number of parameters that can be manipulated, however only a subset of these were changed for these experiments; subsurface scattering, roughness, transmission, and transmission roughness. These were chosen based on observations made looking at the closeups of sand in Figure 2.1. The base color was chosen to match the sand in the HDRI. An IOR for silica was chosen, as silica is always found in regular types of sand (< 95%, otherwise the sand classifies as *Silica Sand*) [12]. This would allow for a known IOR to be used. The IOR of silica is 1.458 [13]. Furthermore, closeups of Silica qualitatively resembles the closeups of sand that these experiments aim to model. This can be seen in Figure 2.10.

The IOR is furthermore used to calculate the specularity parameter for the Principled BSDF



(a) Closeup of silica [14].

(b) Silica at construction site [12].

Figure 2.10: Silica sand at different distances.

shader. According to Blender's manual, specularity should be calculated using the special case of the Fresnel formula seen in Equation 2.1 [11].

$$specular = \frac{\left(\frac{ior-1}{ior+1}\right)^2}{0.08} \quad (2.1)$$

Using the IOR of silica, this gives a specularity of 0.434. These considerations resulted in the values for the BSDF shader used for the sand grains seen in Figure 2.11.

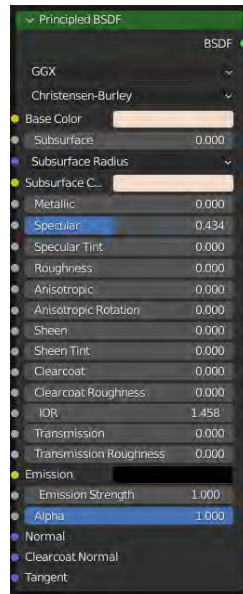


Figure 2.11: Values for principled BSDF used for the sand grains for the pathtracing experiments.

The renders were done for different values of subsurface scattering, roughness, transmission, and transmission roughness. Renders were done for each with values of 0.0, 0.25, 0.50, 0.75 and 1.0. These values were chosen to see the variation in appearance for varying values, however more values could have been chosen further investigate this. When values were updated for one parameter, the remaining were set to value 0.0, except for transmission roughness where transmission was set to 1.0 to see the effects of this parameter. The renders were done using

Blender’s Cycles physically-based path tracer. Cycles allow for rendering different light passes and saving these separately. The light passes include *Diffuse*, *Glossy* and *Transmission*. Diffuse include direct and indirect lighting from diffuse and subsurface BSDFs, excluding color, whereas glossy and transmission include direct and indirect lighting from glossy and transmission BSDFs respectively, excluding color [15]. Direct light refers to light that comes from light sources in the scene that has undergone only a single reflection off or transmission through a surface, whereas indirect light refers to light that has undergone more than one reflection off or transmission through a surface. For instance, specifically for these experiments, diffuse direct refers to light that has come from the HDRI and reflected off a sand grain once before reaching the camera. Indirect diffuse refers to light that has bounced around the scene and has for at least one of the bounces intersected with a sand grain and been shaded by this, before reaching the camera. This is only the case as the sand grains have a diffuse contribution. How a combined image is created in Blender can be seen below. For these experiments, the multiplication with color has been left out of the results for the individual passes.

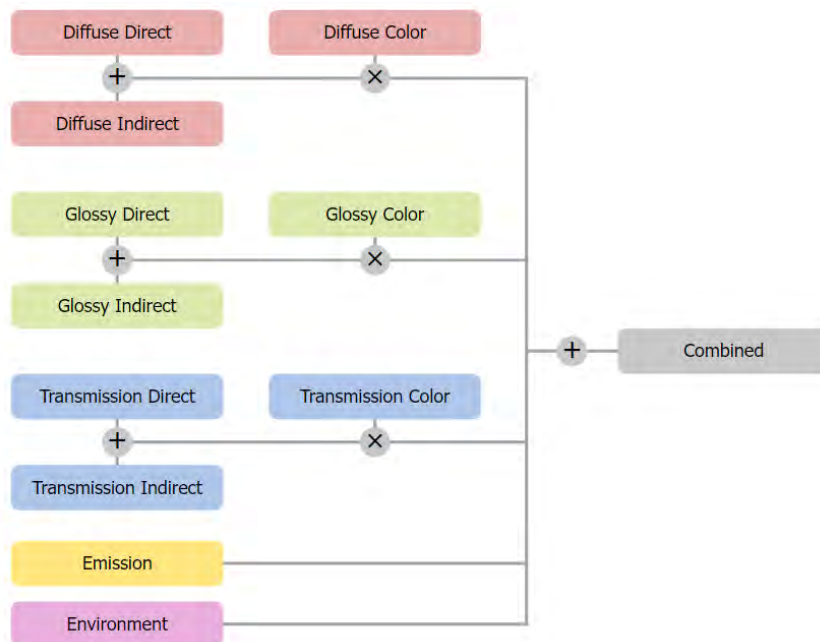


Figure 2.12: Construction of combined image in Blender’s cycles [15].

The results will be presented in order of manipulated properties; subsurface scattering, roughness, transmission, and finally transmission roughness for full transmission. In Blender, subsurface scattering is controlled by a parameter that blends between diffuse and subsurface scattering contributions and by a radius that defines how far the light can travel into the object. For the experiments concerning subsurface scattering, the former was set to 1.0, meaning subsurface scattering was weighted fully, and the subsurface scattering radius was instead multiplied by the values 0.0, 0.25, 0.50, 0.75 and 1.0, meaning the light could travel further and further. For varying transmission roughness, specular roughness was set to the same values as this physically correct. For each of the properties, the results are shown for a single sand grain,

a sphere of sand grains, and wavy planes of sand grains. Throughout the results, the sphere and wavy planes are sometimes described as the aggregate sand objects, as they consist of a collection of individual sand grains. Along with combined renders of each, the render results from the lighting pass that is affected by the varied property is likewise presented. Combined renders and lighting pass refers to the components presented in Figure 2.12. This is specifically the diffuse pass for subsurface scattering (*Diffuse Direct* plus *Diffuse Indirect* excluding *Diffuse Color*), the glossy pass for roughness (*Glossy Direct* plus *Glossy Indirect* excluding *Glossy Color*), and the transmissive pass for transmission and transmission roughness (*Transmission Direct* plus *Transmission Indirect* excluding *Transmission Color*). Finally, details are presented that highlight areas that most clearly show the effects of manipulating the properties.

In Figure 2.13, results for manipulating values for subsurface scattering can be seen. A different subsurface radius was chosen for each of the three geometries (single grain, sphere, and wavy plane), as the radius depend on the size of the geometry in Blender. Values from 0 – 1 refers to a multiplication of the radius for the geometry, i.e. if the radius was set to 0.5, a subsurface scattering value of 0.5 would result in the radius being set to 0.25. This means results from left to right depicts increasing subsurface radii. The results of subsurface scattering are most noticeable for individual grains. Here, areas that are in shadow when no subsurface scattering is used become increasingly brighter as the value of subsurface scattering increases. This is due to the fact that the light rays are scattered in the object and exits at a different location than where they entered, illuminating areas in shadows. Looking at the combined results for the aggregate sand grain objects, the effects are less pronounced. When inspecting the objects closely at the most illuminated areas, the changes can be seen and the sand grains have a milky appearance, and overall the aggregate objects looks darker. The latter is most likely due to absorption happening when subsurface scattering is used. It can be hard to estimate the effects of subsurface scattering at macro scale. Figure 2.14 shows a detailed view for of the wavy plane when the subsurface radius is multiplied by 1 (Figure 2.14a) and when no subsurface scattering (Figure 2.14b) is used. These figures show that when subsurface scattering is used, the diffuse reflectance is less pronounced and more of the environment is reflected towards the observer from every point in the sand grains. This makes sense as the light rays are scattered through the object at another point, meaning points on the surface will have contributions from more locations around the scene. As explained, when light scatters in the sand grains when using subsurface scattering, more light is absorbed. This explains why the results appear darker than for when no subsurface scattering is used. Noticeably, the sand grains appear like more realistic sand grains than when no subsurface scattering is used.

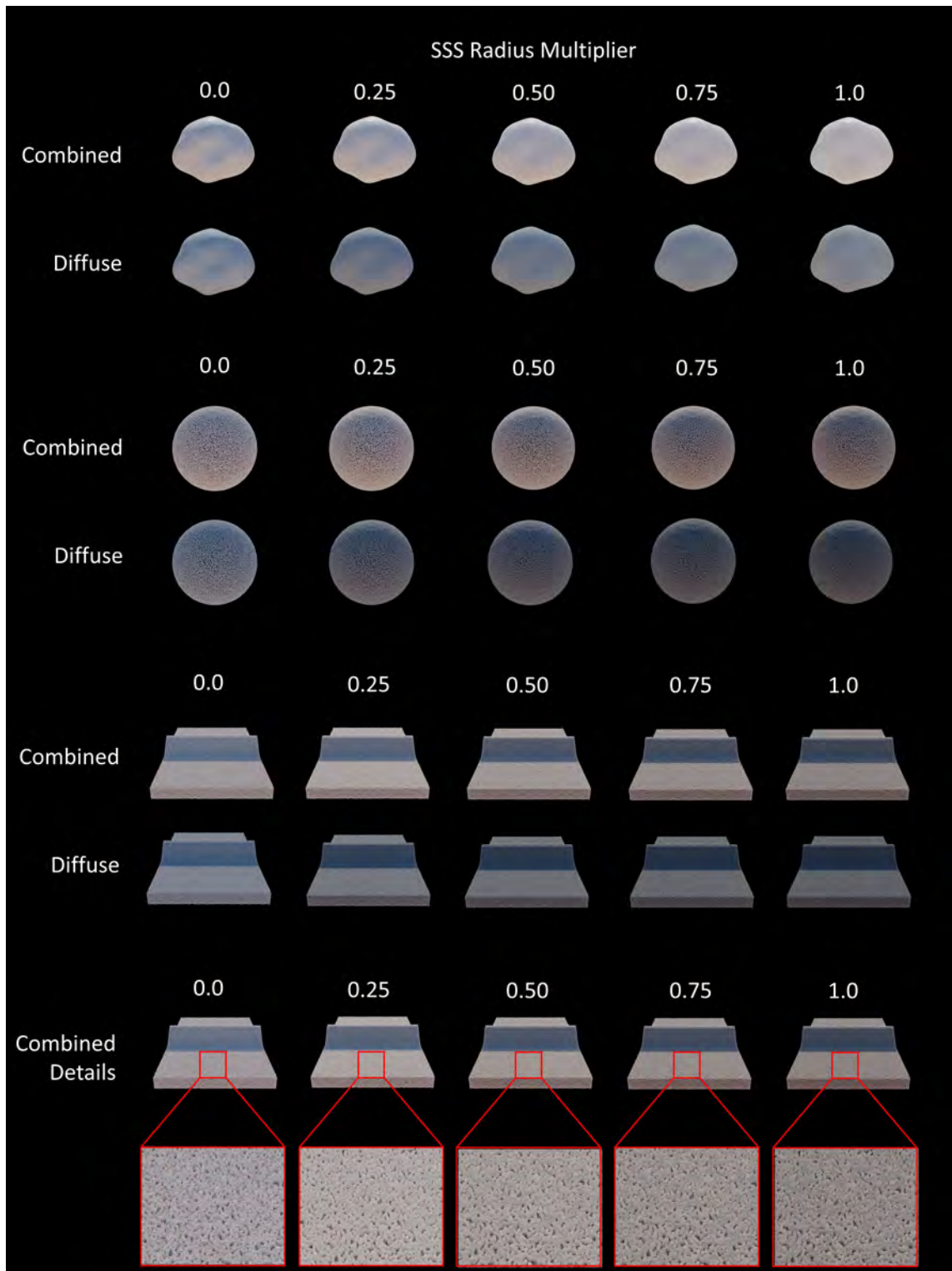


Figure 2.13: Results for changing the subsurface scattering radius in the principled BSDF shader. Values 0.0, 0.25, 0.50, 0.75 and 1.0 were multiplied on the chosen subsurface radius. **Row 1 and 2 (from top):** render results for individual sand grains, combined and diffuse pass respectively. **Row 3 and 4:** render results for a sphere of sand, combined and diffuse pass respectively. **Row 5 and 6:** render results for wavy plane of sand, combined and diffuse pass respectively. **Row 7:** detailed view of row 5. The effects can most clearly be seen for individual sand grains that get a more milky appearance and areas in shadow are increasingly illuminated, where at macro scale the combined results become darker due to increased absorption when values of subsurface scattering increases.

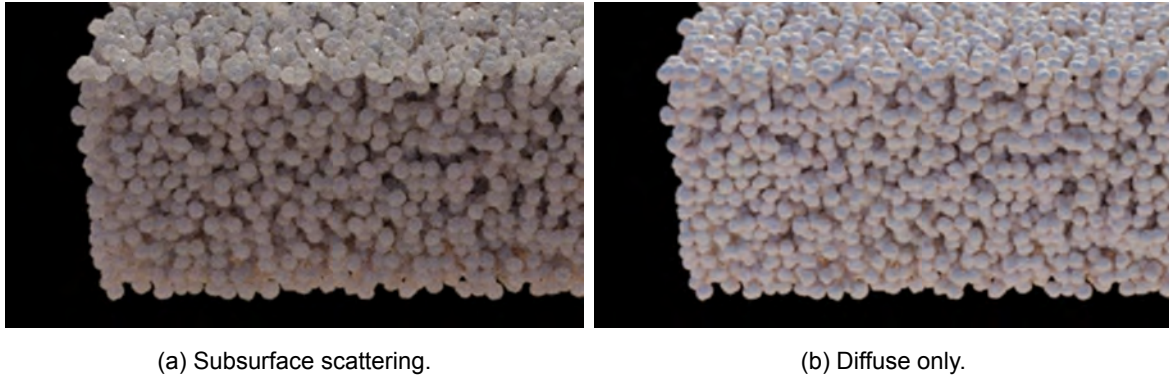


Figure 2.14: Detailed comparison for when the subsurface scattering radius is multiplied by 1 and when no subsurface scattering is used. More light from around the scene contributes to the final color of every point on the sand grain when subsurface scattering is used, since the light is scattered around the grains and exits at another point. Furthermore, more light is absorbed when subsurface scattering is used, resulting in a darker appearance compared to using diffuse only.

Looking at the results of varying roughness in Figure 2.15, specifically at the results in the glossy pass for individual grains, it can be seen as roughness increases, the lobe of reflected light increases and the light is more spread out until it appears almost diffuse for when the roughness value is set to 1.0. When the reflection lobe is very small for low roughness, glints can be observed at macro level, looking at the glossy passes for the sphere and wavy plane. This makes sense, as the presence of glints require small areas on the surface to reflect distinctively more light to the observer than their closest surroundings. When the roughness is very small for individual sand grains, the main light source is reflected off the individual grains at a much smaller area, and the chance of the main light source to be reflected off the sand grain to the eyes of the observer therefore becomes smaller. As roughness increases and the reflected light is more spread out on individual sand grains, the specular reflectance on individual grains approaches the macro level reflectance. For aggregate sand objects, this results in the reflectance approaching the Fresnel reflectance contribution that would be noticeable for uniform surfaces. When the value of roughness is 1.0 and the reflectance on individual grains appears diffuse, the strength of the reflectance at macro level decreases.

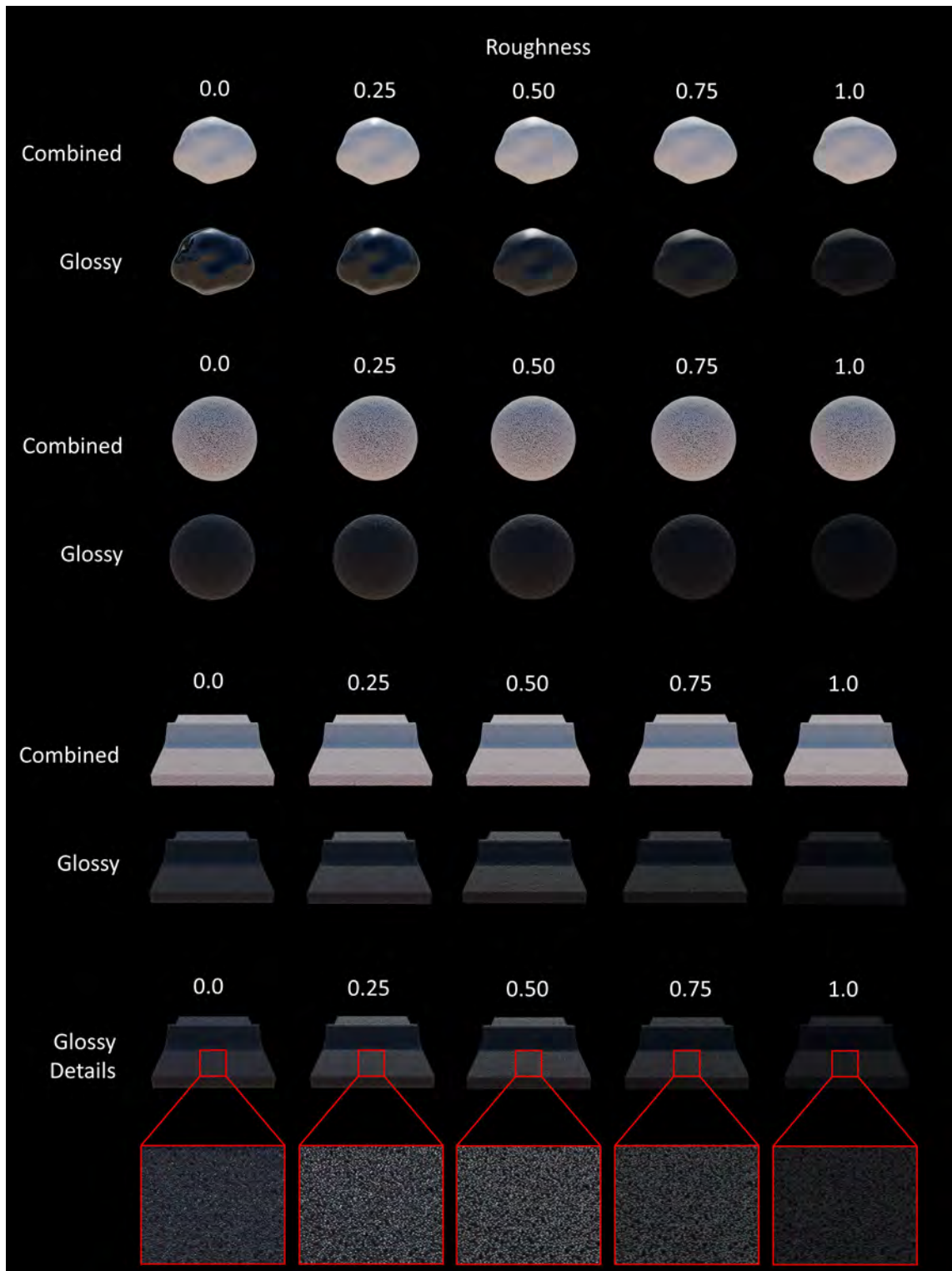


Figure 2.15: Results for changing the roughness parameter in the principled BSDF shader for values of 0.0, 0.25, 0.50, 0.75 and 1.0. **Row 1 and 2 (from top):** render results for individual sand grains, combined and glossy pass respectively. **Row 3 and 4:** render results for a sphere of sand, combined and glossy pass respectively. **Row 5 and 6:** render results for wavy plane of sand, combined and glossy pass respectively. **Row 7:** detailed view of row 6. For low values of roughness, glints can be observed at macro level, and as roughness increases, the reflected light from individual grains is more spread out and at macro level this approaches the reflectance of uniform surfaces. When roughness is set to 1.0, the reflectance appears diffuse.

The results for varying transmission can be observed in Figure 2.16. It should be noted, that setting the transmission value to 1.0 completely removes the diffuse pass in Blender and explains why the combined results of this is distinctively different to the rest. As transmission increases, individual sand grains approaches glass. Looking at the transmission pass for the sphere, it can be seen that most transmitted light can be observed around the edges of the sphere. These areas have a smaller density of sand grains, allowing more light from the main light source to travel through. This is also especially apparent looking at the combined details for the wavy plane in the bottom row, where very concentrated light is refracted through the sand at low density areas at the edge of the aggregate object. The transmitted light can also travel through sand grains across illuminated surface and reach the observer, which adds an approximately uniform contribution of transmitted light across the surface. This can be seen across the surface of the wavy plane geometry. Overall, the macro appearance become darker, most likely a result from absorption of light rays as they travel through the sand grains.

Looking at the results of varying transmission roughness for $transmission = 1.0$ in Figure 2.17, similar effects as could be observed for the glossy pass for varying roughness (Figure 2.15) can be seen. As roughness increases, the light is transmitted through individual grains in a larger lobe, making transmission contribution at micro and macro levels less pronounced. The overall contribution becomes significantly darker as the roughness increases.

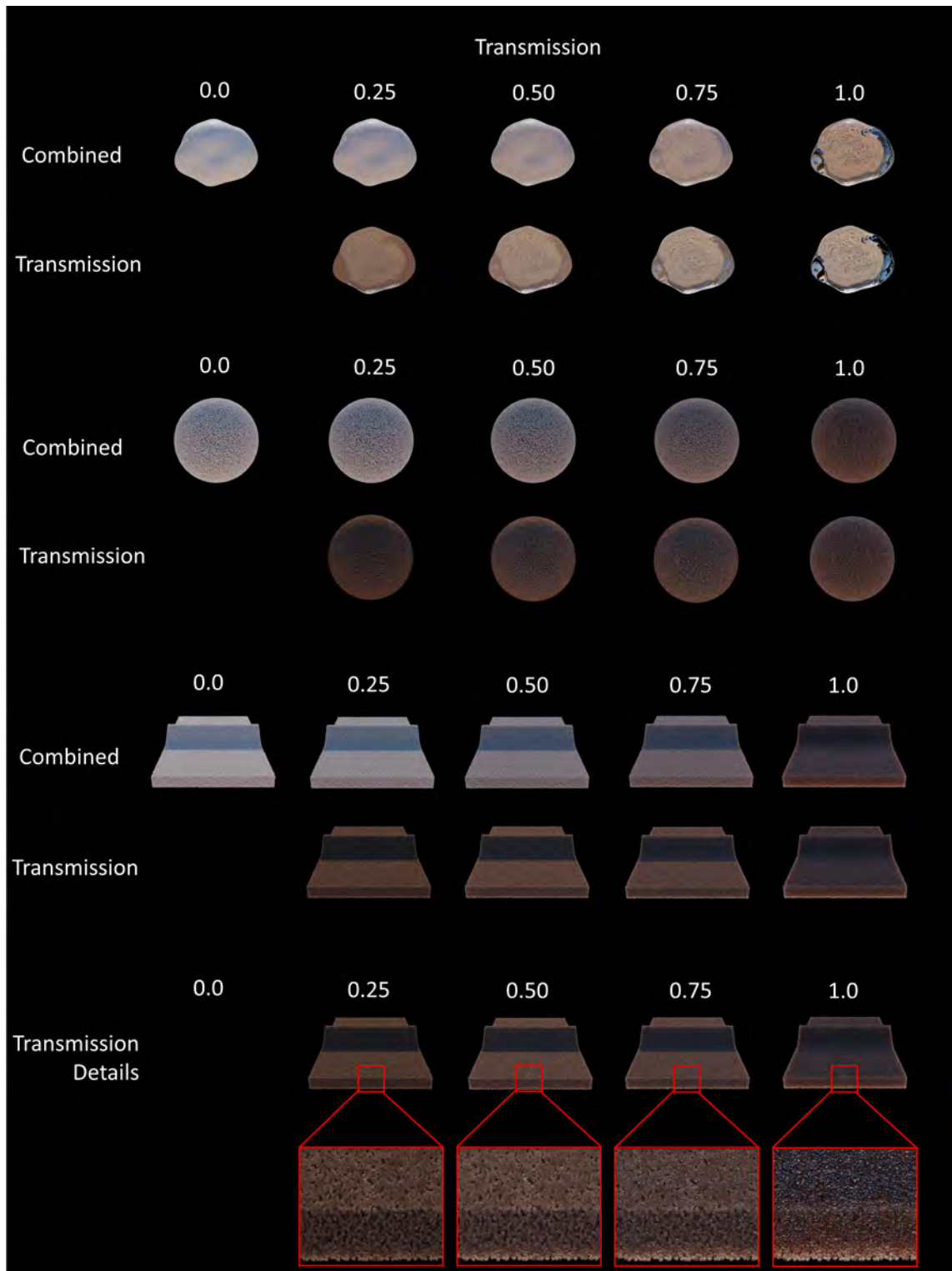


Figure 2.16: Results for changing the transmission parameter in the principled BSDF shader for values of 0.0, 0.25, 0.50, 0.75 and 1.0. **Row 1 and 2 (from top):** render results for individual sand grains, combined and transmission pass respectively. **Row 3 and 4:** render results for a sphere of sand, combined and transmission pass respectively. **Row 5 and 6:** render results for wavy plane of sand, combined and transmission pass respectively. **Row 7:** detailed view of row 6. For increasing transmission, individual grains approach glass while for aggregate objects, concentrated light is transmitted to the observer at low density areas such as the edges of the objects as well as through sand grains at lit areas across surfaces. Absorption makes the results for aggregate objects increasingly darker.

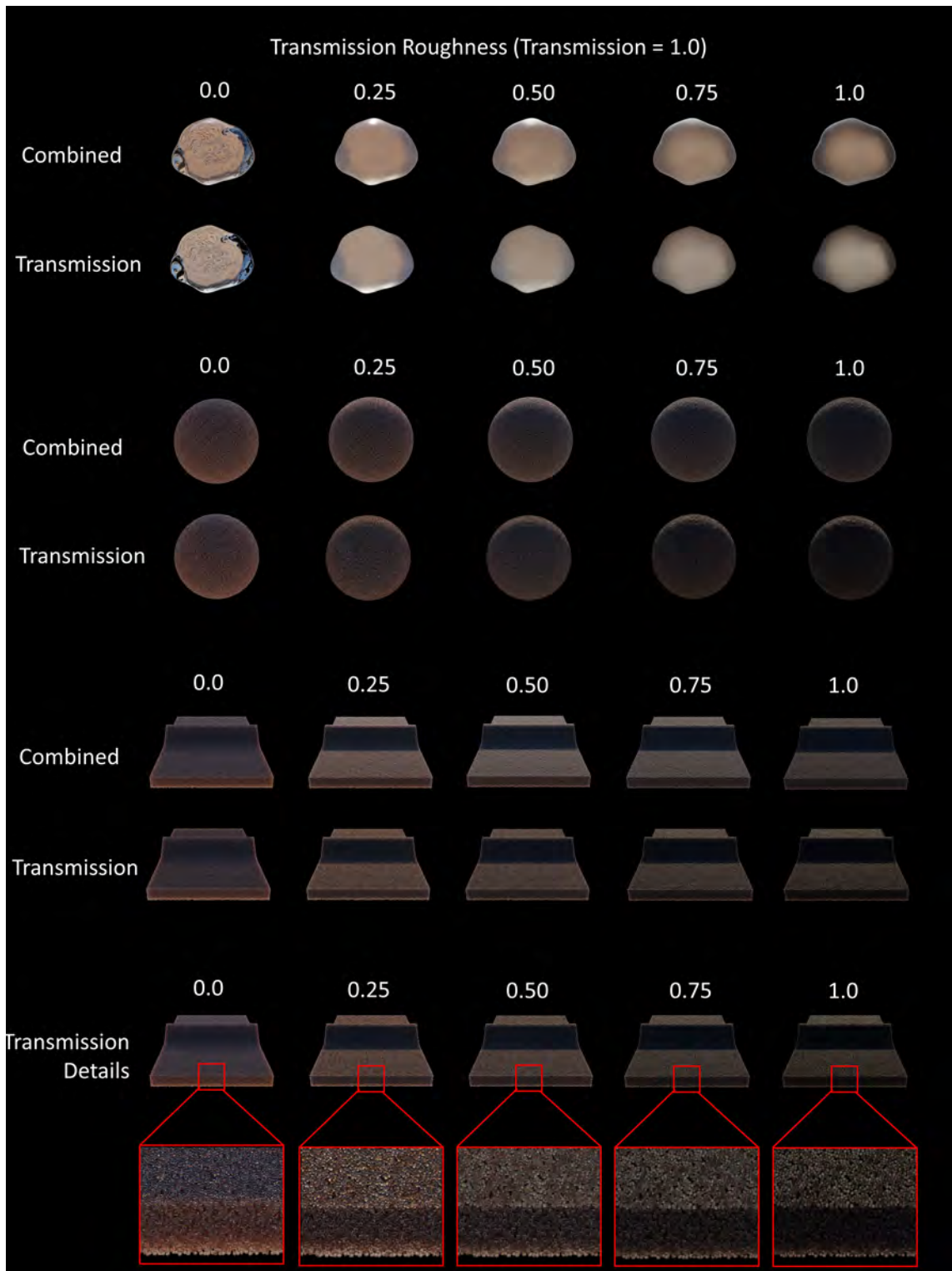


Figure 2.17: Results for changing the transmission roughness and roughness parameter in the principled BSDF shader for values of 0.0, 0.25, 0.50, 0.75 and 1.0 for *transmission* = 1.0. **Row 1 and 2 (from top):** render results for individual sand grains, combined and transmission pass respectively. **Row 3 and 4:** render results for a sphere of sand, combined and transmission pass respectively. **Row 5 and 6:** render results for wavy plane of sand, combined and transmission pass respectively. **Row 7:** detailed view of row 6. The results show that as roughness increases, the light is transmitted through individual grains in a larger transmission lobe, which affects how pronounced the transmission contribution is at macro level.

The results presented in this section provides an overview over how the different material parameters of individual grains influence the appearance of the overall collection of sand grains. However, these experiments are in no way exhaustive, and conducting more experiments could give a very different perspective. For instance, the position of the camera relative to the objects and the light source could have been varied, and the fact that it was not is biased, as the appearance of the materials and therefore the described effects depend heavily on this. Furthermore, the experiments were conducted with identical sand grains, and therefore effects of altering distributions of different types of grains for the aggregate objects cannot be estimated. The experiments could have included altering sizes, shapes, material properties etc. of the sand grains. However, this was deemed outside the scope of this project. Another bias is the fact that the sand grains have been distributed on the objects with a considerable distance between each, as the geometry nodes and Poisson disc distribution did not allow for more closely packed grains. This has an effect on the overall macro level appearance. The reason Blender, a premade pathtracer, was chosen was to acquire results quicker. However, it is worth noting that any pathtracer could have been chosen and the choice of using an already implemented pathtracer means there is less control over the process compared to implementing one from scratch, and there can be limitations in terms of acquiring the results necessary specifically for this type of project.

2.3 Analysis Conclusion

This section illustrate the necessary computer graphics aspects to consider when creating a solution that should model the observations made during the appearance study and offline rendering experiments under the constraints of real-time local illumination models. Some of the most essential limitations that must be dealt with for a real-time solution are listed below. The list is in no way exhaustive, however present the most critical considerations at this point of the project.

1. **Geometric representations.** To achieve highly realistic results, renderings of sand must be made using explicit sand grain geometry, however this is not feasible for a real-time solution. Typically, even at very close distances, hundreds and thousands of sand grains could be present in a rendered image which is excessive to represent real-time. Not being able to represent explicit geometry and explicit stacking and positioning of grains in relation to each other, i.e. the porosity of the sand, introduces challenges for how certain appearance elements should be represented. Looking at the close up of sand grains and the offline rendering results, the stacking of sand grains has a very distinct effect on the appearance of sand. Some sand grains lay occluded by others, while others are fully illuminated directly affecting how granular the sand appears.
2. **Scales.** This project has so far considered micro and macro scale, micro scale referring to the distribution of individual sand grains on the surface of sand and macro scale referring to the surface of sand as a whole. However, micro and macro scale in principle depends

on the current observation scale. For instance, inspecting a single sand grain, micro scale could refer to its distribution of facets and macro scale to the sand grain itself. If observing the dunes of sand from a very large distances, such as looking down at a desert when flying in a plane, micro scale could refer to individual dune orientations and macro to the sand dunes as a whole. This project will continuously refer to the distribution of individual sand grains on a surface of sand as the micro scale of the granular material, and the surface itself as the macro scale of the granular material, and will attempt to model the two. Specifically, micro normals of the surface mesh will refer to the normals of individual sand grains, whereas macro normals refer to the normals of the surface mesh.

3. **Pathtracing.** As seen in the offline rendering results and the results by Meng et al. [3], pathtracing allows for achieving highly realistic results, however is not feasible for real-time solutions. This introduces challenges, as it is no longer possible to trace light through individual sand grains and letting this directly influence the render result at macro scale. The pathtracing experiments provides an understanding of how the material properties at micro scale affect the appearance at macro scale. By utilizing the knowledge of the link between the two, letting the real-time solution of this project model the observations made at macro scale in turn models the micro scale properties. Despite having knowledge of this link, without information about explicit light paths this project would still be challenged when attempting to approximate the macro scale appearance of the pathtracing results, and investigations must be made for how optimal approximations can be made without information about light paths.
4. **Shading.** It was observed that sand grains are not strictly diffuse, specular, transmissive etc., but often a combination of all. A solution should attempt to account for the distributions of each to more accurately model the properties of sand.
5. **Sand grain properties distribution and granularity.** Looking at the appearance study, apart from the porosity of the sand, the distribution of sand grain material properties has a great influence on the granularity of the sand. The sand grains varied in size, shape, diffuseness, specularity, etc. Varying sand grains in size and shape across the surface in real-time can introduce challenges. Without representing the surface by explicit geometry in layers, it can be difficult to ensure a sand grain is always sampled across the surface. A simplification could be to utilize the same geometry and size for each sand grain and ensuring each point of the surface mesh belongs to a sand grain. Varying how diffuse, specular, and transmissive each grain is on the surface of the sand in real-time is likewise not feasible at the current time. The pathtracing experiments did provide a link between material properties at micro scale and the appearance of sand at macro scale, however it was not experimented with introducing differences in material properties between grains, and therefore there is no pathtracing frame of reference to inspire a real-time solution. It is important to find a solution that ensures granularity even when some of the aspects that usually makes the sand appear granular cannot be accurately modelled.

6. **Sampling.** The appearance study showed that at increasing distances, the granular appearance of the sand decreases. This can be explained by the fact that the greater the distance, the more sand grains influence the pixel in the image. For a rendering approach to model this, a great number of samples should be made for each pixel at distances when more sand grains influence the results of each pixel. However, multi-sampling is not optimal for a real-time implementation. A solution should attempt to solve this issue and introduce a method that models the effect of multisampling without the use of it.
7. **Lighting.** The analysis made apparent that only considering a single directional light source will not be adequate for modelling the observations. The results depend greatly on light from the entire environment, and a real-time solution should include environment light contributions to make better approximations of reality.
8. **Distances.** Looking at images of real sand, the appearance varies greatly across different distances to the observer. Specifically, four distances were considered and the appearance at each distance differs considerably from the rest. The videos of sand shows that a seamless and smooth appearance transition happens in real life. For a real-time solution that does not utilize multi-sampling, the distances might have to be defined and handled individually which also requires a smooth transition method to be developed. This demands considerations how many distances are necessary to consider, how to represent each, and how to accurately transition between them. Considering the applications of the implementation of this project, having highly realistic representations at very close distances is, compared to other distances, not critically important. It is furthermore the distance that can result in the highest cost on render time if represented accurately. The player will at most times be observing the sand from standing or crouching height, meaning representations of sand at these distances and toward the horizon should likely be attended to mostly.

The methods of this project are inspired by the knowledge gathered in the appearance study and offline rendering experiments and will attempt to model the observations made while dealing with the real-time challenges listed in this section. The following section presents related work to further inspire the solution of this project.

3 Litterature Review

In this section, related research will be presented and analyzed. An offline rendering technique for multi-scale modeling and rendering of granular materials will first be presented. Following this, three real-time rendering methods will be presented. These deal with modelling granular and porous materials using a BRDF, modelling specular granularity by sampling procedurally generated normal distributions at micro scale, and taking an artistic approach to rendering sand in real-time for a game.

3.1 Multi-Scale Modeling and Rendering of Granular Materials

The approach by Meng et al. [3] was briefly described in the introduction, but this project can benefit from further understanding their offline approach for rendering highly realistic granular materials. Meng et al. [3] proposed a method for offline rendering granular materials at multiple scales. Their method allows the user to specify the shape of the aggregate granular mesh, the packing rate and scale of the grains, i.e. the density, as well as the geometry and material properties of the individual grains and the distribution of different types of grains. This allows for realistic rendering of many types of granular materials with distinct packing rates and individual grain properties, such as sand, snow, different spices, etc. The approach first uses explicit path tracing (EPT) of the individual grains. After the rays have undergone a certain number of scattering events, volumetric path tracing (VPT) is used to approximate the scattering through the remaining grains of the aggregate material. Finally, diffuse approximation (DA) is used as an approximation of the contribution of any further interactions. To allow for rendering the granular materials at multiple scales, they uses an automatic switching criterion to switch between the three methods (EPT, VPT, and DA). The process can be seen in Figure 3.1.

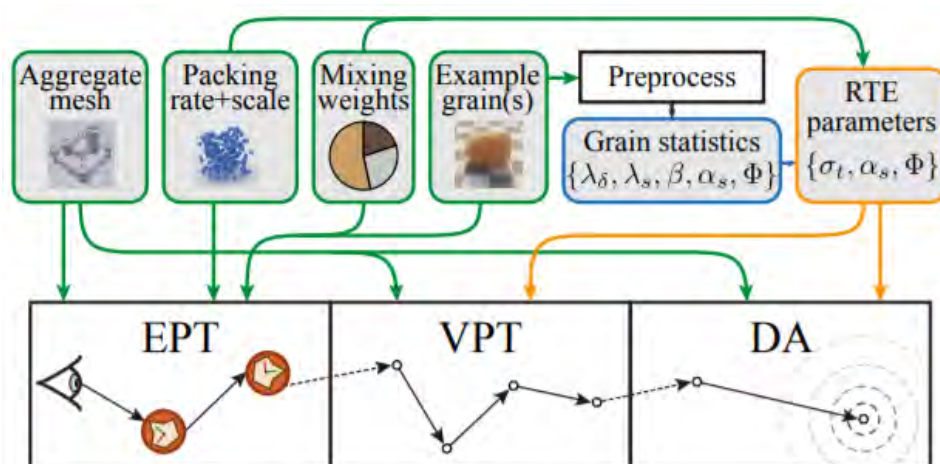


Figure 3.1: Approach for rendering of granular materials at multiple scales [3].

This approach results in highly realistic renderings at multiple scales, e.g. when observing the

objects closely, the individual grains can be discerned and at increasingly larger distances, the method convincingly models the combination of multiple grains as an aggregate granular material.

While this approach produces very realistic results it is not appropriate for real-time usage. However, it describes components that influences the appearance of a granular material, such as overall shape of the aggregate material, packing density, and distributions of types of grains. A real-time solution could be inspired by these components in which approximations are made that resemble their contributions to the final rendering results.

3.2 An Analytic BRDF for Materials with Spherical Lambertian Scatterers

D'Eon et al. [4] developed an analytic BRDF for porous materials that attempts to alleviate the limitations present for existing rough diffuse BRDFs, such as Oren-Nayar. Oren-Nayar and other popular techniques are not fully suitable for porous and granular micro geometry, as they are based on a diffuse random height-field. There are limits as to how spiky height fields can be while remaining physically plausible, and the range of plausible roughnesses that can be presented are therefore restricted. D'Eon et al. instead uses a volumetric approach to derive their BRDF. The volumetric approach models the micro surface structure as a collection of particles that are both absorbing and scattering within a volume. They do so by modelling the scattering particles as Lambertian spheres. Their BRDF furthermore focuses on permitting easy artist control by having only a single editable parameter of the BRDF, namely the albedo of the Lambertian spheres. As can be seen in Figure 3.2, they compare the appearance of an array of Lambertian micro geometries with varying packing density and scale with a fixed diffuse albedo to height-field based BRDFs with varying roughness. It shows that the height-field bases techniques accurately resembles dense packings (bottom rows), however for sparse media they fall short and artifacts can be observed.

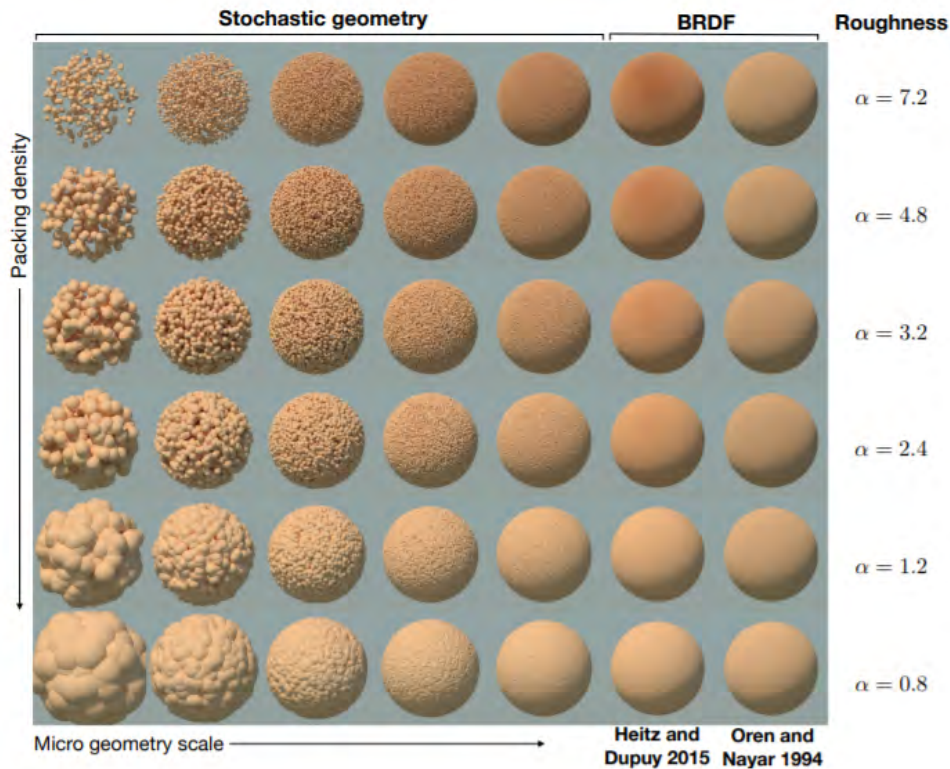


Figure 3.2: Results by d'Eon et al. [4] from comparing height-field based techniques (two most right columns) with an array of Lambertian spheres of varying packing density and scale and fixed albedo. The results showed that while the typical height-based techniques convincingly represent dense granular media (bottom rows) they cannot accurately represent sparse media (top rows), in which more collisions occur on average before rays escape the aggregate medium.

Their experiments show that the appearance of their BRDF exhibits strong back scattering and saturation effects, which cannot be reproduced by techniques such as Oren-Nayar. However, while their approach provides results that exhibit effects that the typical random height field based techniques cannot reproduce, there are several effects that cannot be achieved using their analytical BRDF. For instance, they do not alleviate the limitation to how many plausible roughnesses can be represented height-field methods, as their BRDF does not support roughness which limits the potential of the technique. Furthermore, their BRDF currently does not support the effects of varying micro geometry scale and packing density that can be seen in Figure 3.2, however they hope to support this through their BRDF in future work.

3.3 Real-Time Rendering of Procedural Multiscale Materials

Zirr et al. [16] proposed a real-time procedural method for rendering materials having irregular micro surface structure at multiple scales. The approach provides user control over appearance at all scales of the material and allows to model several complex material effects, among other glints. Firstly, the method introduces a bi-scale noise distribution function (NDF). The NDF is defined on an arbitrary patch, in this case specifically by a cell on a regular grid in object uv

space. The density of the grid can be controlled by the user and essentially models the density of micro details per unit area in uv space. To simulate multi-scale micro details, they utilize a hierarchy of grid cells on top of the first level grid cells, each level being twice as dense as the previous, modelling different LODs. The same biscale NDF is evaluated at each grid level which constructs an infinite number of scales. The model is evaluated by estimating the pixel covered area of micro details reflecting light. This introduces challenges, as the pixel footprint can cover an arbitrary sized region inside the hierarchy. This issue is dealt with by evaluating the pixel footprint on each grid and two grid levels are chosen based on this; one is the closest coarser level for the projected pixel granularity where the other is the closest finer level. The shading is evaluated by blending between the shading results of each grid cell, which is a similar process to anisotropic filtering, which helps ensuring anti-aliasing. Besides the density of micro details, the user can control the roughness of the material at macro scale and at micro scale, the variation in micro detail scale, and finally the overall intensity of the material. Results of their method can be seen for sand and snow in Figure 3.3 below.

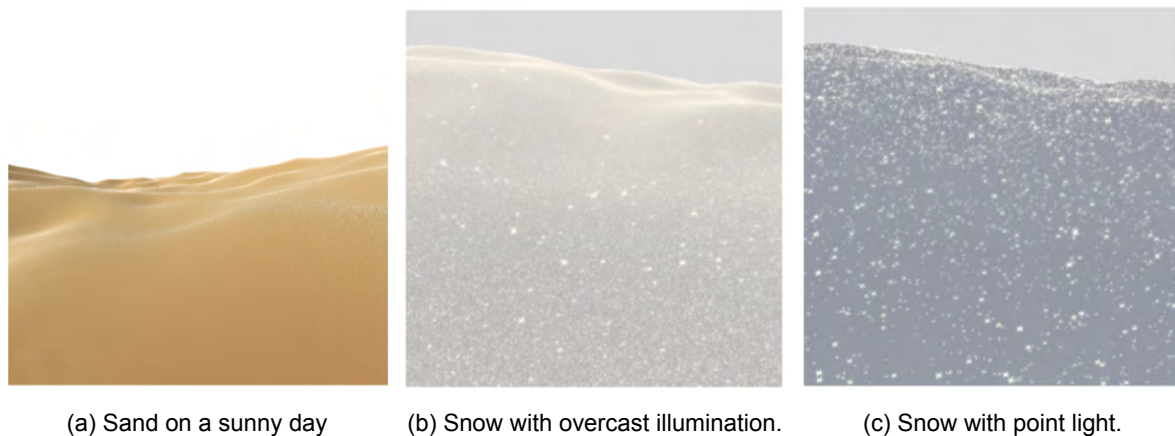


Figure 3.3: Rendering of sand and snow using the approach by [16].

This implementation provides a good example of rendering of granular materials in real-time at multiple scales, where an NDF is evaluated at multiple levels of grid cells in object texture space, ensures anti-aliasing at multiple levels. Using the pixel footprint for evaluating the normal information is a good approach as this scales with distances and correctly can be used to represent the amount of differentiating normal information for a pixel. At shorter distances, the normal variation is less varied where at greater distances the normal information becomes more varied as a pixel covers more granules. Furthermore, the approach is based on physically accurate models while allowing for artistic user control that makes the results deviate from a strictly physically accurate one. Their approach can especially inspire the implementation of specular components in this project, however it models specular micro facet surfaces and lacks the extension of diffuse and transmissive characteristics observed during the appearance study. Another drawback to their method is that the granularity that can be seen in their results that stems strictly from the specular component. This can be a drawback for their method, as

granularity does not directly extend to materials that are not specular.

3.4 Real-Time Sand Rendering in Journey

A real-time rendering technique for granular materials relevant for this project is the one used in the video game *Journey*, produced by *thatgamecompany* in 2012 [17]. The game features a character that takes on a journey through a vast world of sand and snow. In his breakdown of the rendering of sand at GDC13 [18], John Edwards describes how the company achieved the iconic visuals of sand in the game. It should be noted that the goal of the implementation of *Journey* was not to necessarily be realistic but to capture a certain artistic feel and vision that the creators had. Therefore, compared to other work discussed in this section, this solution is more artistically driven and is not directly attempting to model physically plausible sand, providing a different perspective to real-time rendering of granular materials.

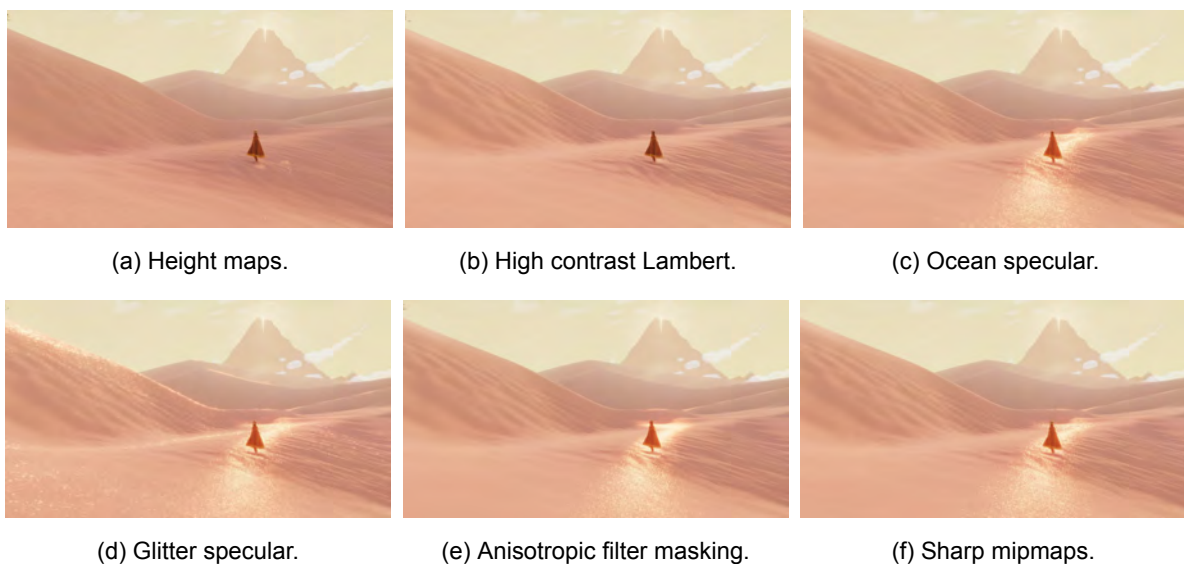


Figure 3.4: Sand rendering in *Journey* [18]. From top left to bottom right; height maps for ripple effects, high contrast Lambertian diffuse, ocean specular effect, glitter effects, anisotropic filter masking, and sharp normal mipmaps. The images from top left to bottom right include the effects in the preceding images. The purpose of the effects were not to recreate physically accurate sand, instead they were developed with the focus of providing a specific user experience envisioned by *thatgamecompany*, providing a different perspective on how a process look for the development of rendering solution for a video game application.

Figure 3.4 shows the different effects presented by Edwards in his talk [18]. Firstly, geometric details are added to the surface by introducing height map textures at different frequencies, modelling ripples effect in the sand that can also be observed in real sand (Figure 3.4a). For the diffuse part of their shader, they utilized a customized, high-contrast version of Lambert that alleviated some of the issues that occurred using standard Lambert, while remaining cheap compared to using Oren-Nayar (Figure 3.4b). For specularity, an ocean specular effect was implemented (Figure 3.4c). This effect is not physically accurate for sand and was not observed

by the team in their appearance study of sand, however it gives the desired effect of the character flowing through an *ocean of sand*, as was the vision of the team. On top of the ocean specular, *glitter* effects were implemented in which a noisy normal map was used for creating specular highlights in which light was reflected off individual grains of sand into the eyes of the observer (Figure 3.4d). This phenomenon has been referred to as *glints* in other parts of this report. This was done by utilizing a specular shader that was dependent on the view direction. This implementation in some cases resulted in undesired visual effects at specific parts of the surface. This can be seen at the top of the dune to the left in the image in Figure 3.4d, where the glints are *smearred out* over too many pixels, which was not the intention of the team. To combat this, the team used an anisotropic filtering mask to check where it is safe to draw the glints and where their anisotropic filter would fail and glints should therefore not be used (Figure 3.4e). Finally, they used a mipmap sharpening effect for the noisy normal map, resulting in a more grainy appearance even at a distance (Figure 3.4f). They observed real sand to be appearing smoother the greater the distance to their camera, however the creators wanted rather than to model this physical phenomenon to maintain the graininess feeling as much as possible even at a distance.

The implementation of sand in Journey shows a process for visual development that does not approach it from a physically accurate experience but rather from a human experience perspective, providing an alternative perspective to how the process could be tackled. It inspires how a solution can be made that takes into account the creative freedom that can be desired for applications such as video games.

3.4.1 Conclusion

The literature study showed several ways of approaching the rendering of granular materials. While methods such as the one presented by Meng et al. [3] offer highly realistic results the methods are not suitable for real-time usage as they require storing explicit granular geometry and utilizing pathtracing techniques to obtain the realistic results. On the other hand, micro surface structures can be represented by a BRDF, such as for the method by d'Eon et al. [4], which is highly suitable for real-time usage. While the technique presented by d'Eon [4] provides a solution that alleviates some limitations of typical diffuse micro reflection models, it does not alleviate all and since it utilizes a diffuse BRDF it cannot on its own represent all the characteristics discovered in the appearance study. It furthermore does not model roughness, a limitation in comparison to other micro facet reflection models, such as Oren-Nayar [5]. Zirr et al. [16] presents a solution that models specularity for multiple scale materials in real-time, allowing representation of complex effects such as glints. The method uses the uv-texture of the mesh to define grids of normal distributions which are sampled using the pixel footprint. The implementation by Zirr et al. [16] has a number of user controllable variables, allowing for artist directability. Like the method by d'Eon et al. [4], the method does not model all characteristics observed during the appearance study. The rendering of sand in Journey [17] also does not account for all characteristics of sand observed in the appearance study,

however presents an approach in which an implementation is guided by the user-experience rather than physical phenomenon alone. While this approach results in an engaging and appealing experience for the user, it compromises realism. For instance, they purposely diminish the physical phenomenon arising for granular materials at greater distances to the observer by enforcing granularity where this would naturally not be observable. Each of the methods presented in this section can in their own way inspire a solution for this project, and they each deal with some of the characteristics observed in the appearance study, however neither of them on their own capture all observed properties. The methods of this project is motivated by taking inspiration from the presented methods while attempting to capture the qualities of sand discussed in the analysis.

4 Methods

In this section, the methods of this project will be presented. They are based on the requirements posed by Playdead, the knowledge gained in the analysis, as well as the research gathered in the literature study. They attempt to approximate the observations made in the analysis, using the offline pathtracing results as a frame of reference, while attempting to deal with the issues identified and explained in Section 2.3.

4.1 Micro Normals

This section deals with how to represent the surface at micro-level, i.e. how to acquire normals of individual sand grains on a surface mesh. As described, representing the mesh with complex geometric sand grain representations in multiple layers is not feasible for a real-time solution, and the very close up distance that would require these representations most likely will not be critical in a game. Therefore, an alternative to complex micro-scale representations should be utilized. A simple representation could be inspired by Zirr et al. [16] and represent the surface as a grid and associate points in object space across the surface with a grid cell, where a grid cell represents a sand grain. Each grid cell could be associated with specific normal information. Since transitioning to particle representation is needed, the grid cell could furthermore be associated with simple geometry information. This would allow particles to detach from the surface while retaining the same geometry and start position, rotation and scale. The used geometry should be very simple to decrease computational demands. Generating pseudo random numbers seeded by the position of the given fragment on the surface allows associating points on the surface with individual sand grains. The random numbers can then be used to define the rotation of the sand grain and hereby its normal information.

Jarzyński et al. [19] evaluated a range of hash functions for random number quality in their paper about *Hash Functions for GPU rendering*, among other a PCG3D variant, proving to be a relatively fast hash while having good quality, suitable for real-time graphics applications. PCG3D takes a 3D input and produces a 3D output through a series of multiply-and-add operations. In their paper, Jarzyński et al. presents an implementation of PCG3D, which can be seen in Listing 4.1.

```
1 uint3 pcg3d(uint3 v)
2 {
3   v = v * 1664525u + 1013904223u;
4   v.x += v.y*v.z; v.y += v.z*v.x; v.z += v.x*v.y;
5   v ^= v >> 16u;
6   v.x += v.y*v.z; v.y += v.z*v.x; v.z += v.x*v.y;
7   return v;
8 }
```

Listing 4.1: PCG3D implementation [19].

The output values lie in the ranges of $[-1, 1]$. By seeding this function by the position of the fragment in object space, the mesh can be represented by a grid of different random numbers for each grid cell, where a grid cell can be used to represent a sand grain. This idea can be seen illustrated in Figure 4.1. The figure shows a plane and how generating pseudo random numbers based on the fragment position in object space results in a grid like structure, where each grid cell is associated with a set of three random numbers. The set of random numbers have been transformed to the range $[0, 1]$ and illustrated as RGB colors. Using the *pcg3d* function, the resolution of the grid can be controlled by scaling the input vector v . The higher the multiplication factor on v , the higher the resolution. This determines the size of each sand grain and the number of sand grains across the surface.

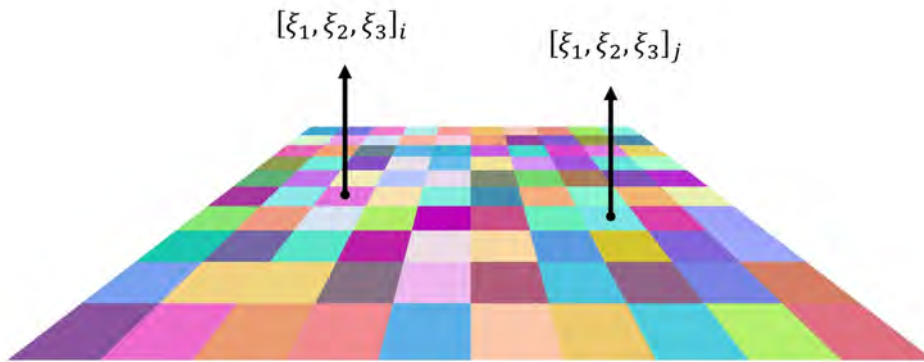


Figure 4.1: Illustration of the grid of pseudo random numbers generated by PCG3D when seeded by the position of a fragment in object space. The random values are scaled from range $[-1, 1]$ to $[0, 1]$ and shown as RGB colors. Each grid cell is associated with a different set of random values, shown by the random value vectors $[\xi_1, \xi_2, \xi_3]_i$ for object position i and $[\xi_1, \xi_2, \xi_3]_j$ for object position j .

As described, simple geometry should be used to represent the sand grains. In reality, no sand grain has the exact same shape, however for a real-time solution that is also suitable for particle representation, the sand grains could be associated with the same geometry to decrease computational demand. One of the simplest geometries that still model facets on a sand grain and allow for effects such as glints, compared to using e.g. using spheres, is the cube. The cube can be represented by only six unit normals. A rotation for a grid cell, i.e. sand grain, can be represented by a unit normal which can be randomly sampled on a unit sphere. Dutré [20] describes how a random point (x, y, z) on a sphere can be generated with two random numbers ξ_1 and ξ_2 in the interval $[0, 1]$. Given a sphere (c_x, c_y, c_z, R) where c is the center coordinates of the sphere and R is the radius, the random point can be found using the formula presented in Equation 4.1.

$$\begin{aligned}
\phi &= 2\pi\xi_1 \\
\theta &= \arccos(1 - 2\xi_2) \\
x &= c_x + 2R\cos(\phi)\sqrt{\xi_2(1 - \xi_2)} \\
y &= c_y + 2R\sin(\phi)\sqrt{\xi_2(1 - \xi_2)} \\
z &= c_z + R(1 - 2\xi_2)
\end{aligned} \tag{4.1}$$

As this is used to find a random rotation for the sand grain, a unit sphere with center at $c = (0, 0, 0)$ and radius $R = 1$ can be used, and the equations can therefore be simplified. The random point (x, y, z) on a unit sphere can be used to represent a direction and therefore a rotation for the grid cell. Hereby, each grid cell is represented by pseudo random numbers seeded by the position of the fragment in object space, normals of a unit cube, and a rotation from sampling on a unit sphere using the pseudo random numbers. This is illustrated in Figure 4.2.

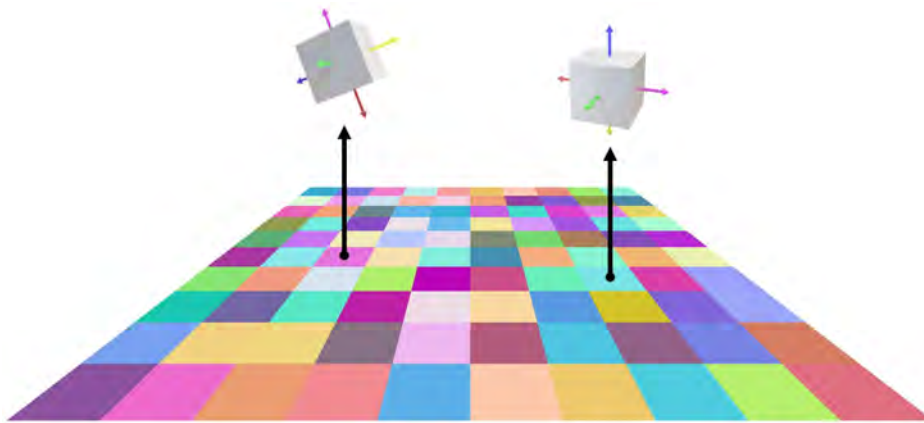


Figure 4.2: Illustration of how each grid cell is represented by cube normals and a random rotation which is generated using pseudo random numbers seeded by the position of the fragment in object space. The colors of the cube normal vectors in the figure are merely selected for illustration purposes and do not represent actual directions. Both illustrated cubes have the same normal vectors and colors but are rotated differently, signifying different rotations for each grid cell.

From here, intersection tests could be performed to check which point on the cube the view direction intersects. However, by explicitly modelling the cube in the grid cell introduces challenges. It has to be ensured that cases in which we do not intersect with the cube are handled. A simpler approach could be to simply check which side of the cube align with the view direction and using the normal of this as the micro normal for the grid cell. The sides that are sampled as micro normals for the grid cells are the ones that are most similar to the view direction, i.e. the vector towards the camera. To check which side of the rotated cube aligns with the view direction, the view direction should be transformed to the object space of the cube. Frisvad [21] presents an approach to build an orthonormal basis from a unity 3D vector, which can be seen in Listing 4.2.


```

1 void frivsvad (const Vec3f & n , Vec3f & b1 , Vec3f & b2 )
2 {
3   if(n.z < -0.9999999 f) // Handle the singularity
4   {
5     b1 = Vec3f ( 0.0 f , -1.0 f , 0.0 f );
6     b2 = Vec3f ( -1.0 f , 0.0 f , 0.0 f );
7     return ;
8   }
9   const float a = 1.0 f / (1.0 f + n.z );
10  const float b = -n.x*n.y*a ;
11  b1 = Vec3f (1.0 f - n.x*n.x*a , b , -n.x );
12  b2 = Vec3f (b , 1.0 f - n.y*n.y*a , -n.y );
13 }

```

Listing 4.2: Finding an orthonormal basis from a 3D unit vector ([21])

The orthonormal basis b_1, b_2, n can be used to create an inverse model transformation matrix for the rotation normal and hereby transform the view direction to object space. In this case, for the function in Listing 4.2, n should be the point sampled on the sphere in Equation 4.1. Once the view direction vector has been transformed to object space, the most aligned cube normal can be found by checking which of the cube normals has the largest dot product with the view direction. This is illustrated in Figure 4.3. Depending on the size of the grid cell and the rotation of the cube, multiple cube normals can be sampled across the grid cell, given that the view direction varies enough across the area of the grid cell.

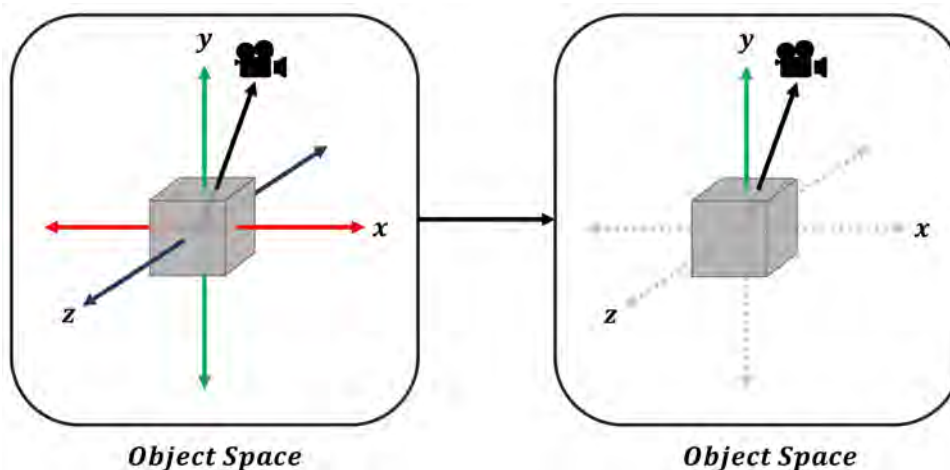


Figure 4.3: The view direction vector is transformed to object space using the inverse transformation matrix of the sampled rotation direction of the grid cell. The unit cube normals in object space align perfectly with the positive and negative directions of the three axes x, y and z . To find which cube normal aligns the most with the view direction, the unit cube normal that has the largest result when dotted with the view direction is chosen. In this figure, the transformed view direction has the largest dot product with the unit cube normal in the positive y -direction, and therefore this is chosen.

The sampled cube normals are then transformed to world space using the transformation matrix

of the rotation direction of the grid cell, i.e. the random point sampled on the sphere. The sampled cube normals will be referred to as *micro normals* in this project, as they are used to represent individual sand grains at a micro level of the surface. *Macro normals* instead refer to the normals at macro level, i.e. the normals of the surface mesh.

Overall, this approach for sampling micro normals provides a simple and efficient approach to modelling the micro structure of the surface, however several crude approximations are made in the process, which are important to consider. Firstly, using a cube as sand geometry is not physically accurate which will have an influence on the result of the rendering of sand. Furthermore, sampling normals based only on alignment with the view direction without regard for intersection with the explicit sand geometry has its limitations. By using the normals that are most aligned to the view direction, there will be normals that are never sampled, such as normals at glancing angles. It also means, that when observing the sand grains very closely the facets of the sand grains are not visible, as the view direction does not vary enough across such a small area that the sand grain covers, and it is highly likely that the same normal will be sampled across all the points covered by the sand grain. However, while having the potential of modelling more realistic sand geometry, explicitly modelling geometry and checking for intersections provides its own challenges. For example, if the grain is modelled explicitly, situations in which the sand grain is not intersected has to be handled. A way to solve this could be to make certain the geometry is always larger than the grid cell. A third approach could be to use a procedurally generated normal map for each sand grain. This method like the rest has its benefits and limitations. The normal information could vary across the sand grain and hereby more realistically model the facets present in a single sand grain and make the micro representation more realistic, however again it must be ensured that the normals are not pointing away from the view direction or the macro normal direction. Additionally, it has to account for the normal information that is not seen by the camera in the case that the grain represented on the surface detaches and transitions to particle form, something that is handled using the cubes representation that models geometry and rotation. As can be seen, each approach has its advantages and limitations, and while the latter two will most likely provide more realistic results if the cases where they fall short are handled, the approach that has been proposed in this section is quite efficient, simple and easy to extend; the same information is stored for all sand and few, simple operations must be done to test which normal to use, and it can be further extended to include more complex geometry by introducing more normals to check for. It furthermore makes transitioning to particle representation simple, and since often thousands of particles can be present in the same image at the same time, it requires very simple and efficient information storage for each, which makes the cubes approach a good candidate compared to the rest of the presented approaches. Using this approach however means the porosity of sand cannot be represented geometrically for the surface. Instead, this should be approximated in the shading process of the sand grains. This and remaining considerations related to shading will be discussed in the following section.

4.2 Shading

The appearance study suggested a number of shading contributions necessary to represent sand. This include diffuse reflectance, specularly, and transmission, the latter two dependent on values of roughness. This section covers sampling of light and shading of the surface, attempting to model the observed properties of sand and dealing with the challenges listed in Section 2.3. The variables presented in Table 4.2 will be used extensively throughout the sections and have been listed in the table for ease of understanding. All variables are defined in world space for the calculations throughout.

Abbreviation	Definition
x	Fragment position
m	Micro normal
n	Macro normal
ω_o	Direction toward observer
ω_i	Direction toward light
$:=$	Signifier for updating a value in Equations

4.2.1 Lighting

The analysis showed the need for considering light from the entire environment and not just an idealized light source for realistic results. For the real-time implementation, an HDR image can be used for environment lighting like for the offline pathtracing technique. The optimal solution would be to utilize the different intensity levels stored in the HDR to set the intensity of the light source, however the solution of this project approximates this by using an HDR for retrieving light at a uniform intensity, and a separate directional light source to model the sun in the HDR with an adjustable intensity. The incident light at surface point x with normal n from direction ω_i will throughout the following sections be referred to as $L_i(x, \omega_i)$, whereas the incident light from the environment at surface point x from random directions within hemisphere Ω will be referred to as $L_{sky}^\Omega(x, \Omega)$. The approach for sampling the environment is inspired by Bærentzen et al. [22], and can be seen in Equation 4.2.

$$L_{sky}^\Omega(x, \Omega) = \frac{1}{N} \sum_{j=1}^N L_{sky}(x, \omega_{i,j}) \quad (4.2)$$

where N is the number of directions used for sampling the environment, $L_{sky}(x, \omega_{i,j})$ is the incident light from the environment from direction $\omega_{i,j}$. The direction $\omega_{i,j}$ is sampled on a cosine-weighted hemisphere around macro normal n , using two uniform random variables that differ for different values of j .

4.2.2 Sand Grain Colors and Porosity

As described, at current time the link between distributions of sand grain properties and the macro surface appearance is not understood. Therefore, the granularity of sand cannot be

modelled through having sand grains differ in how diffuse, specular, or transmissive they are or how subsurface scattering events takes place within them. A way to model the granularity of the sand could be to instead utilize different colors for the sand grains. This property was also observed in the appearance study, where even sand appearing very uniform at far distances consists of grains with distinct colors. To make color assignment random, a color can be chosen for each sand grain using the generated pseudo random numbers. To allow artist directability, a finite set of controllable colors can be used. The random numbers generated are 3 values in range $[-1, 1]$. By scaling these to $[0, 1]$, they can be made binary and hereby be used to extract a color value from the set. First, the random values are made binary as shown in Equation 4.3.

$$\xi_{binary}^i = \begin{cases} 0 & \text{if } \xi_i < 0.5 \\ 1 & \text{otherwise} \end{cases} \quad (4.3)$$

Where ξ_i is a random number in range $[0, 1]$, and the conversion is done for each of the three random values generated. Then the three binary random numbers are used to retrieve an index for getting a color value. If a set of 8 colors is used, the conversion to an index can be seen in Equation 4.4.

$$\rho_m = C[4\xi_{binary}^1 + 2\xi_{binary}^2 + \xi_{binary}^3] \quad (4.4)$$

where ρ_m is the color of the sand grain where the notation of m refers to micro normal, C is the set of 8 color values, and ξ_{binary}^1 , ξ_{binary}^2 and ξ_{binary}^3 are the three binary pseudo random numbers calculated using Equation 4.3.

Another property relating to granularity that was observed was the porosity of the sand. The methods for this project does not allow for representing this geometrically, and instead a shading approach can be used. Essentially, a sand grain can be occluded by other sand grains to varying degrees. Some might be visible but occluded by their neighbors so that less light can reach them, while others might be stacked on top, fully illuminated. A simple approximation of this could be to multiply the sand grain with an intensity value, modelling the level of occlusion of the sand grain. Like the colors of the sand grains, the illumination factor for each sand grain could be determined using random numbers. The observations made during the appearance study suggests having a range of possible illumination values that are not too low, i.e. multiplying the color of the sand grain with 0 is probably not reasonable. This is due to the fact that, if environment light is used, grains visible to the camera will be illuminated to some degree. A minimum intensity value I_{min} can be determined using the formula in Equation 4.5.

$$I_{min} = e^{(-P)} \quad (4.5)$$

where P is the porosity factor and I_{min} is the minimum intensity value a sand grain can have. Values of P should range between $[0, 1]$. The higher the value of P , the more porous, i.e. the

larger intensity differences, as I_{min} is used in conjunction with the random values to calculate the intensity value of the sand grain in the way that is presented in Equation 4.6. Looking at real sand, the porosity factor declines quickly with increasing distances to the observer. To model this observation, P should be divided by the distance $dist$ between the camera and the point being shaded in world space. The final calculation of I_{min} can be seen in Equation 4.6.

$$I_{min} = e^{(-\frac{P}{dist})} \quad (4.6)$$

Equation 4.7 shows how the intensity value of the sand grain $\rho_m^{intensity}$ is calculated using I_{min} .

$$\rho_m^{intensity} = I_{min} + (1 - (\xi_3(1 - I_{min}) + I_{min})) \quad (4.7)$$

where $\rho_m^{intensity}$ models the amount of light reaching the sand grain in the grid cell, I_{min} is the minimum intensity value, and ξ_3 is one of the three generated random values in range $[0, 1]$. By multiplying this value to the final shading result for the sand grain, each sand grain is assigned a random intensity within the range $[I_{min}, 1]$, modelling the occlusion effect of sand grains. Scaling ξ_3 to be in range $[I_{min}, 1]$ (right-most part of the equation) ensures a uniform distribution of intensities in range $[I_{min}, 1]$.

When looking at the sand at macro level, ρ_m is no longer relevant. A representation of the sand as a whole must be made. At macro level, a multitude of sand grains cover each pixel. A good approximation could be to utilize the average color of the sand grains. This inspired the macro color of the sand ρ_n , seen in Equation 4.8.

$$\rho_n = \frac{1}{8} \sum_{i=1}^8 C[i] \quad (4.8)$$

where ρ_n is the macro surface sand color, which is an average of the set C of sand color values. Aside from macro surface sand color, the average macro intensity should be found. This is simply the mean value of the possible intensity values as seen in Equation 4.8.

$$\rho_n^{intensity} = I_{min} + \frac{1 - I_{min}}{2} \quad (4.9)$$

where $\rho_n^{intensity}$ is the macro intensity value. $I_{min} + \frac{1 - I_{min}}{2}$ represents the average intensity of the sand grains when using the approach for calculating the micro intensity values calculated in Equation 4.7.

Using a set of user defined color values gives the user complete control over the colors used for the sand. However, having to define all of these could be cumbersome. Instead, an approach that lets the user define less values and letting interpolations between these values take place behind the scenes could make the process easier for the user, and using interpolations would

provide more variation in color. However, this in turns means the user is in less control over the final appearance. Furthermore, interpolations can result in undesired color values. The approach of representing porosity by an intensity value is not strictly correct, since it does not model the visual occlusion of sand grains, meaning the sand grain being occluded is still fully visible, just less illuminated, which might not always be the case. The reason it cannot model this is that the sand grains are all equally visible as they are defined in a grid structure. Nonetheless, it can provide a method for representing the intensity variation of sand grains that were observed across the surfaces of sand.

The following sections break down how the the diffuse, specular, and transmissive contributions for the sand will be approximated. As described, the link between distributions of shading contributions at micro scale and the appearance at macro scale is not determined. Therefore, the following sections describe the attempt to model the macro scale appearance with the pathtracing results as a frame of reference. This means that instead of using micro normals to shade individual grains and let this decide the appearance at macro scale, the shading will be done using macro normals which will attempt to capture the offline frame of reference. Another reason for this is the fact that the micro normals are separate from the macro normals and based on random rotations, and therefore the macro information is lost if shading is done purely with micro normals. The granularity of the sand grains will be expressed through the varying sand grain colors and porosity factor described in this section. The micro normals will however be used for expressing glints, as this effect is directly influenced by the micro structure of the surface.

4.2.3 Diffuse Reflectance and Subsurface Scattering

The analysis showed that sand has a diffuse contribution which should be considered for the shading of the sand grains. For calculating diffuse reflectance, the light from the environment and intensity of directional light source should be used. For calculating the diffuse shading considering both the skylight and the main directional light source, equations inspired by Bærentzen et al. [22] can be used.

$$L_d(x, \omega_o) = \frac{\rho_m}{\pi} (L_{sky}^\Omega(x, \Omega) + L_i(x, \omega_i) \max(0, n \cdot \omega_i)) \quad (4.10)$$

where $L_d(x, \omega_o)$ is the diffuse reflectance in direction ω_o at surface point x , ρ_m is the color of the sand grain, $L_{sky}^\Omega(x, \Omega)$ is the incident environment light, $L_i(x, \omega_i)$ is the incident light from the high intensity light source in direction ω_i . The dot product between macro normal n and ω_i should be multiplied on $L_i(x, \omega_i)$, as this describes how much the light direction aligns with the macro normal n and therefore how illuminated the surface is. Sampling the environment light $L_{sky}^\Omega(x, \Omega)$ on the other hand happens in random rotations in a hemisphere around the macro normal and should therefore not be multiplied by this term. Equation 4.10 assumes that the material is perfectly diffuse. As observed, it is not, but controlling how much $L_d(x, \omega_o)$ contributes to the final shade will be explained in later sections.

When looking at the results for subsurface scattering in the offline rendering results, it was seen that as the subsurface scattering radius increases, shadows become less pronounced on individual sand grains as more light from the environment is scattered through the sand grain and exiting at each point. Furthermore, the aggregate objects overall appear darker, most likely as a result of the light rays being absorbed in the sand grains. This can be revisited for the wavy plane in Figure 4.4. The former observation can be modelled by making the diffuse reflection less normal dependent, i.e. weighting the term $\max(0, n \cdot \omega_i)$ less. The latter observation can be approximated by introducing an extinction factor to model the absorption of light in the sand grains. This can be seen formalized in the following equations.

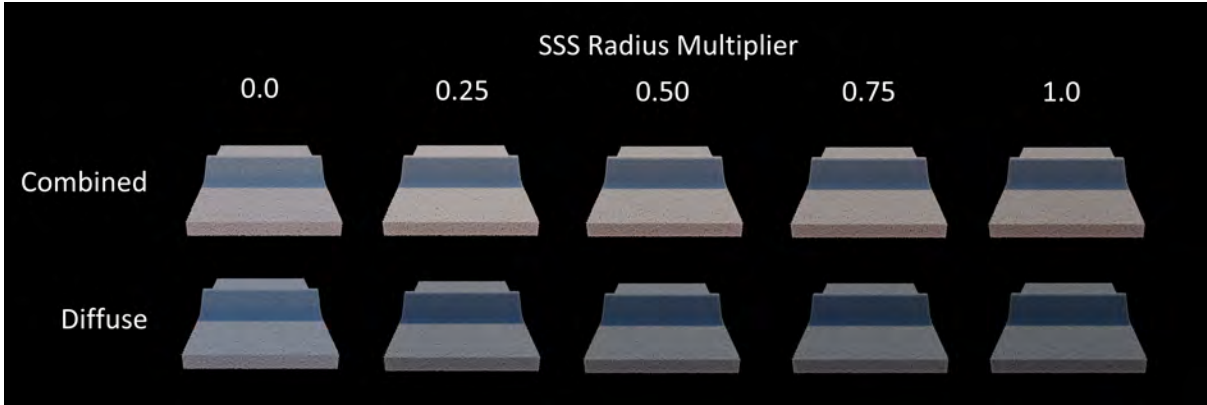


Figure 4.4: Combined and diffuse lighting pass for varying subsurface scattering radius multiplier. These results lay the foundation for modelling the effect of increasing subsurface scattering radius for this project.

$$\begin{aligned}
 L_d(x, \omega_o) &= \frac{\rho_m}{\pi} L_i(x, \omega_i) (f_s + (1 - f_s) \max(0, n \cdot \omega_i)) \\
 L_d(x, \omega_o) &:= L_d(x, \omega_o) + \frac{\rho_m}{\pi} L_{sky}^\Omega(x, \Omega) \\
 L_d(x, \omega_o) &:= f_e L_d(x, \omega_o)
 \end{aligned} \tag{4.11}$$

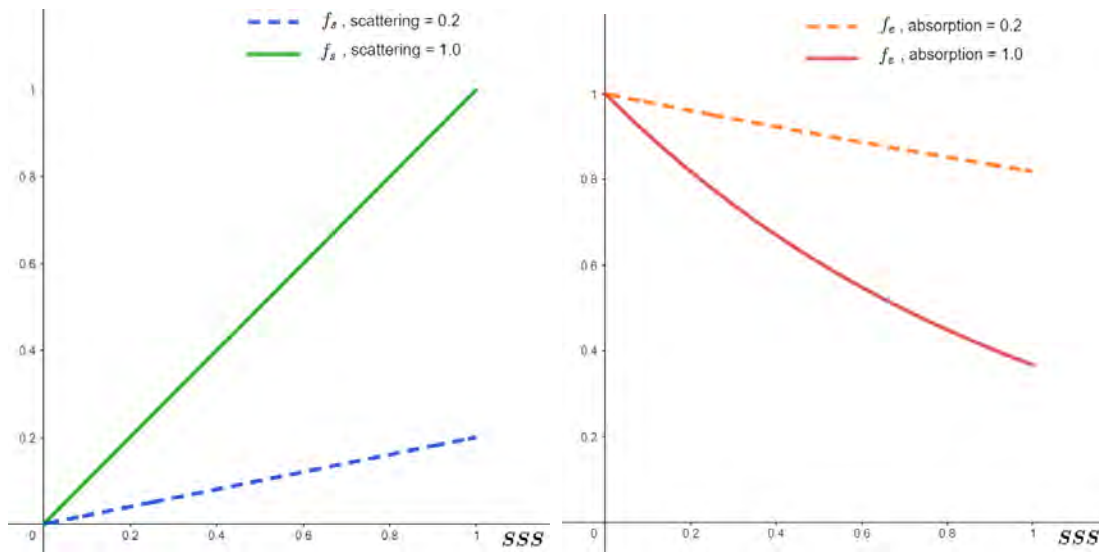
where $:=$ is used to signify updating the diffuse term $L_d(x, \omega_o)$, f_s is the scattering factor, and f_e is the extinction factor, modelling the absorption of light in the sand grains. f_s is used to determine how much the term $\max(0, n \cdot \omega_i)$ is weighted for the final diffuse reflection; the higher the values of f_s , the less dependent on the normal the diffuse reflection becomes. f_e simply controls the intensity of the diffuse reflection, i.e. the inverse of how much light is absorbed. The calculations of f_s can be seen in Equation 4.12.

$$f_s = \text{scattering } SSS \tag{4.12}$$

where SSS is variable that models the subsurface scattering radius multiplier in Blender, and *scattering* controls the amount of scattering, i.e. for high values of *scattering*, less of the term $\max(0, n \cdot \omega_i)$ is weighted and vice versa. Setting *scattering* = 0.5 for instance ensures that even for high values of SSS , the cosine-term is still weighted. *scattering* should be in range $[0, 1]$. The calculations of f_e can be seen in Equation 4.13.

$$f_e = e^{(-\text{absorption } SSS)} \quad (4.13)$$

where *absorption* controls the amount of absorption. Higher values of *absorption* result in less intensity for the final diffuse reflection. Using an exponent to control the extinction factor ensures that even for $SSS = 1$ and *absorption* = 1 not all light is absorbed, resulting in no diffuse reflection. *absorption* should be in range $[0, 1]$. Illustrations of functions f_s and f_e can be seen in Figure 4.5 for values of *scattering*= 0.2 and 1.0 and *absorption*=0.2 and 1.0. Figure 4.5a shows that for increasing values of *scattering*, f_s become larger which in turn weighs the normal dependent term $\max(0, n \cdot \omega_i)$ less in Equation 4.11. Figure 4.5b shows that as *absorption* increases, f_e decreases which results in an overall reduced intensity of the diffuse contribution, given Equation 4.11.



(a) Function f_s for scattering values 0.2 and 1.0. (b) Function f_e for absorption values 0.2 and 1.0.

Figure 4.5: Functions f_s and f_e for scattering and absorption values 0.2 and 1.0. The output of f_s is constrained by the value of scattering. As the value of absorption increases, the output of f_e decreases.

4.2.4 Specular Reflectance

The analysis showed that sand at macro scale has a view dependent specular reflection contribution. This was especially apparent for the tops of dunes at a distance, where more light was reflected from the sky toward the viewer. This can be modelled using Fresnel equations [23] using the macro normal for the surface. This could likewise be observed for the pathtracing results where the Fresnel reflectance became stronger as specular roughness increased for individual sand grains, until the roughness was so high the sand grains appeared diffuse. Another specular component observed both during studies of real sand and the pathtracing experiments were glints. These were observable at macro scale but depend on the micro surface structure, i.e. facets of individual sand grains. Compared to the rest of the shading contributions of this project, the expression for glints will utilize the micro normal direction m . This

section will explain how these two contributions can be implemented for a real-time solution. The results of the specular contribution for the wavy plane can be revisited in Figure 4.6.

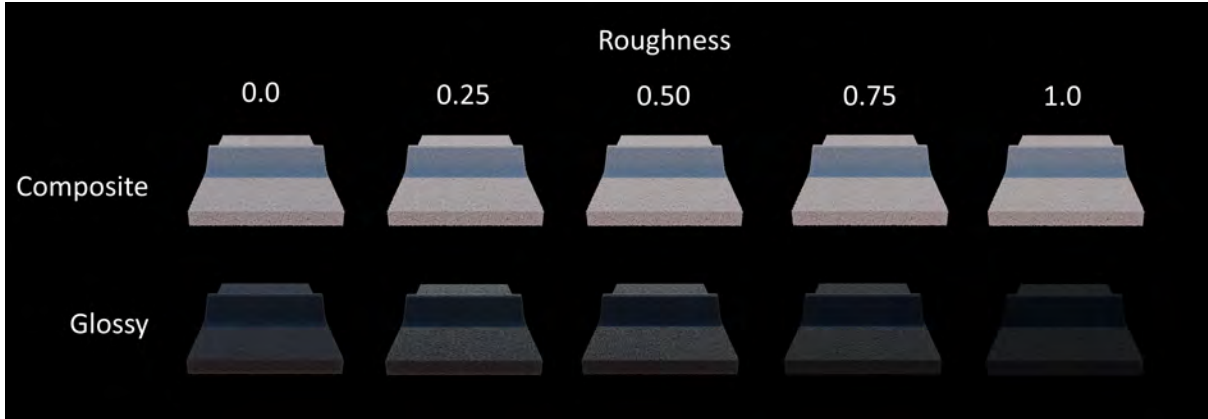


Figure 4.6: Combined and glossy lighting pass for varying roughness. These results lay the foundation for modelling the specular contribution of this project.

Typically for computer graphics applications, Slick's approximation for the Fresnel factor in specular reflection is used [23]. This can be seen in Equations 4.14 and 4.15.

$$f = f_0 + (1 - f_0)(1 - (\omega_h \cdot \omega_o))^5 \quad (4.14)$$

where

$$f_0 = \left(\frac{n_1 - n_2}{n_1 + n_2} \right)^2 \quad (4.15)$$

f is the specular reflection coefficient for surface point x , ω_o is the direction toward the observer, ω_h is the halfway vector between the light direction ω_i and the reflected light direction, calculated using the macro normal n , and n_1 and n_2 are the indices of refraction of the incident medium and the material. The order of the two is arbitrary as the sign of the result is removed when the term is multiplied by itself. The incident IOR is typically air and can be therefore be approximated as 1. For this project, the IOR of Silica could be used for the material as in the pathtracing experiments. The specular reflectance from the Fresnel contribution can then be calculated as seen in Equation 4.16.

$$L_f(x, \omega_o) = f(L_{sky}^\Omega(x, \Omega) + L_i(x, \omega_i) \max(0, n \cdot \omega_i)) \quad (4.16)$$

where $L_f(x, \omega_o)$ is the light reflected towards the observer, f controls how much of the incident light from the sky $L_{sky}^\Omega(x, \Omega)$ and main light source $L_i(x, \omega_i)$ is reflected in the direction towards the observer ω_o . The term $\max(0, n \cdot \omega_i)$ is multiplied on the incident light from the main light source to control the amount of reflection from the main light source depending on how much it aligns with the surface normal.

The pathtracing experiments showed that low specular roughness of sand grains are needed for glints to appear, as light needs to be reflected off a sand grain in a concentrated way. The field trip furthermore showed how glints change as the view direction does. Glints from sand grains closer to the observer appear and disappear more slowly in comparison to the ones from sand grains at a distance, as the view direction varies less for closer sand grains than ones further away. An obvious way for implementing glints would be to use the sampled micro facet normal, as these represent facets on the sand grains, and then evaluating the dot product between the view direction and the reflected light direction to determine the strength of reflection. However, since these are sampled based on view direction alignment, this results in an excessive number of glints. It was therefore deemed necessary to utilize another approach. When observing glints in images or in real life, they seem evenly distributed across a surface of sand. This inspired using a Gaussian Distribution Function (GDF) to control whether a sand grain is able to create glints. A GDF can be controlled using a mean μ and a standard deviation σ [24]. The GDF can be seen in Equation 4.17.

$$GDF(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (4.17)$$

This form of the Gaussian in which division by $\sqrt{2\pi\sigma^2}$ is left out ensures the output of the function does not exceed 1. Setting $x = \xi_x$ where ξ_x is one of the pseudo random numbers in range $[-1, 1]$ and setting $\mu = 0$ will output uniformly distributed values in the range $[0, 1]$. The shape of the curve can then be determined by σ , and determines how many of the grid cells representing sand grains will be associated with high GDF output values close to 1, and how many will be associated with lower values close to 0. Lower values of σ means a narrower distribution, resulting in more low output values, and higher values of σ means a broader distribution, resulting in more high output values. This can be seen looking at Figure 4.7, showing the GDF for $\sigma = 0.1$ and $\sigma = 0.5$, both for $\mu = 0$.

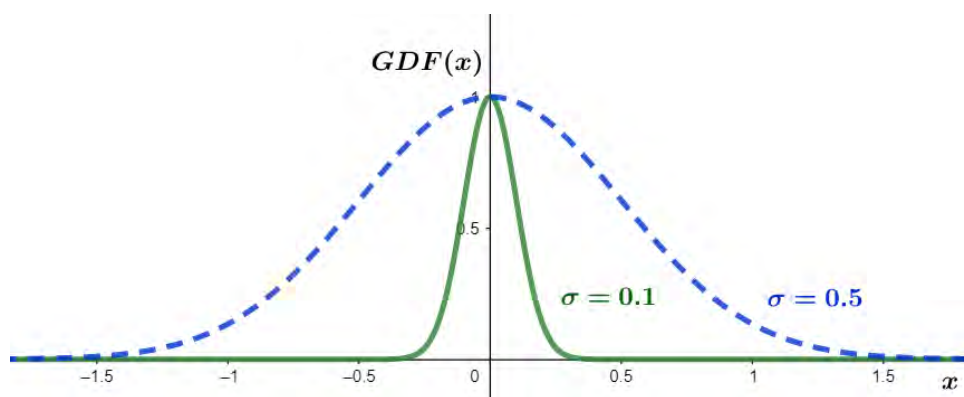


Figure 4.7: Gaussian Distribution Functions for $\sigma = 0.1$ (green solid line) and $\sigma = 0.5$ (blue striped line), based on Equation 4.17. As σ increases, the curve flattens and a larger number of output values will be close to 1.

As described, glints could be modelled by evaluating the dot product between the view direction

and the reflected light direction based on the micro normal, however this resulted in excessive glints. Using the output of the Gaussian, this can be controlled. Multiplying the output of the Gaussian with this dot product ensures that glints can only appear on certain sand grains, controlled by the value of σ , and therefore the excessive number of glints can be avoided. This idea can be seen in Equation 4.18.

$$g = GDF(\xi_x) (\omega_o \cdot (\omega_i - 2.0m(m \cdot \omega_i))) \max(0, n \cdot \omega_i) \quad (4.18)$$

where g represents the strength of the glint for point x , ξ_x is one of the pseudo random number generated for point x , ω_i is the incident light direction, ω_o is the direction towards the observer, and m is the micro normal direction. The term $\omega_i - 2.0m(m \cdot \omega_i)$ is the reflected light direction using the the micro normal. The expression is also multiplied by the term $\max(0, n \cdot \omega_i)$, as glints only model strong reflections from the main light source and should therefore not occur if not illuminated by this. Equation 4.20 show the glints reflection contribution.

$$L_g(x, \omega_o) = L_i(x, \omega_i) g \quad (4.19)$$

$L_g(x, \omega_o)$ is the light reflected in direction ω_o at point x towards the observer, and $L_i(x, \omega_i)$ is the incident light from the main light source in direction ω_i . Since glints depend on the strong intensity of the main light source, the skylight is not added for this expression. It was noticed that the glints reflection was not strong enough to appear like glints observed during the field trip. It was decided to multiply the glints contribution by a high intensity I_{glints} to enhance their effect.

$$L_g(x, \omega_o) := L_g(x, \omega_o) I_{glints} \quad (4.20)$$

This approach is not physically accurate. Multiplying by I_{glints} means that more energy exits at the point than arrives, which is incorrect. Furthermore, controlling glints using a GDF and not simply the micro normals is likewise inaccurate. However, the approach results in very believable glints that behave similarly to the ones observed during the field trip. Another benefit is the fact that since the distribution of glints can be controlled by σ , it allows for artist directability, as the artist can be in control of how pronounced an effect glints should be.

As seen during the pathtracing experiments, the results of the glossy lighting pass were dependent on the roughness of the individual grains. As roughness increases, the reflection lobe of the light becomes larger and the light is reflected more diffusely. As a consequence, the strength of glints decreases and the reflected light appears more uniform across the macroscopic surface, resulting in a strong Fresnel reflectance, until the roughness is so large that the effect of the specular component on the surface is insignificant to the overall appearance. This can be revisited in Figure 4.8. This can be qualitatively modelled for the methods used in

this project by quantifying the strength of the glints and Fresnel effects based on a roughness parameter. This could lead to a function that outputs a strength value based on the roughness value. Table 4.1 qualitatively quantifies how strong the glints and Fresnel reflectance appears on a scale from 0 – 1 (0 being no effect, 1 being the strongest) in Figure 4.8 for varying values of roughness.

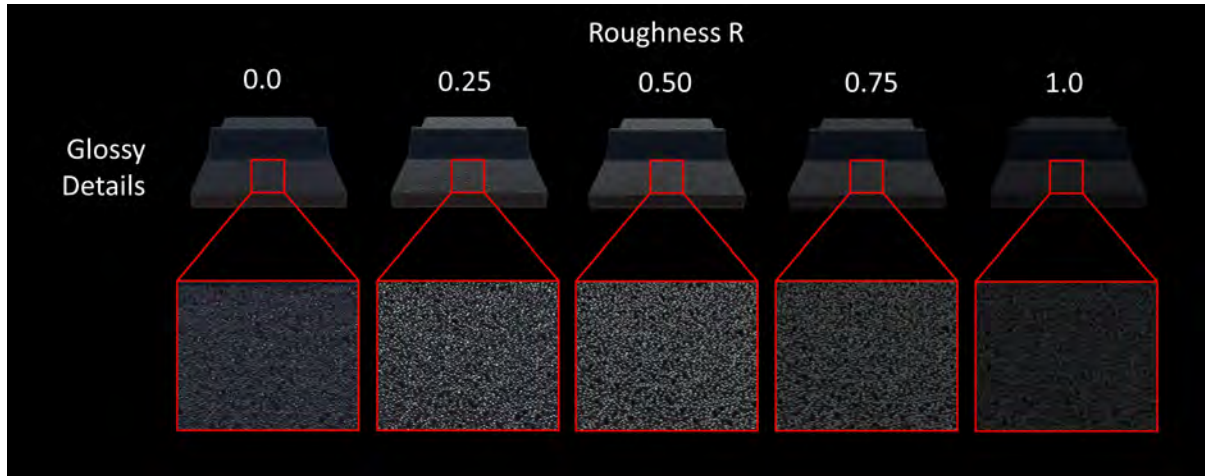


Figure 4.8: Glossy lighting pass details for varying roughness. These results lay the foundation for assessment of the strength of the glints and Fresnel effects given the value of roughness.

R_s	0.0	0.25	0.5	0.75	1.0
G	1.0	0.9	0.25	0.0	0.0
F	0.25	1.0	1.0	0.4	0.1

Table 4.1: Strength G of glints effect and strength F of the Fresnel effect based on specular roughness parameter R_s . The strength values range between 0 – 1, 0 being no effect and 1 corresponding to maximum strength. The values are qualitatively assessed from the results in Figure 4.8.

Looking at the values in Table 4.1, $G(R_s)$ can be approached by a sigmoid function [25]. $F(R_s)$ on the other hand first increases and then decreases, and the slope of the increase is greater than the decrease. This can be approached by a polynomial. Functions $G(R_s)$ and $F(R_s)$ can be seen in Figure 4.9. The equations behind the functions can be seen in Equations 4.21 and 4.22. Note, that $G(R_s)$ does not exactly capture the values in Table 4.1, however is deemed adequate.

$$G(R_s) = \begin{cases} \frac{1}{1+e^{13.5(R_s-0.4)}}, & \text{if } 0 \leq R_s \leq 1 \\ \text{not defined,} & \text{otherwise} \end{cases} \quad (4.21)$$

$$F(R_s) = \begin{cases} \min(1, -20.80R_s^5 + 60R_s^4 - 55.90R_s^3 + 14.55R_s^2 + 2R_s + 0.25), & \text{if } 0 \leq R_s \leq 1 \\ \text{not defined,} & \text{otherwise} \end{cases} \quad (4.22)$$

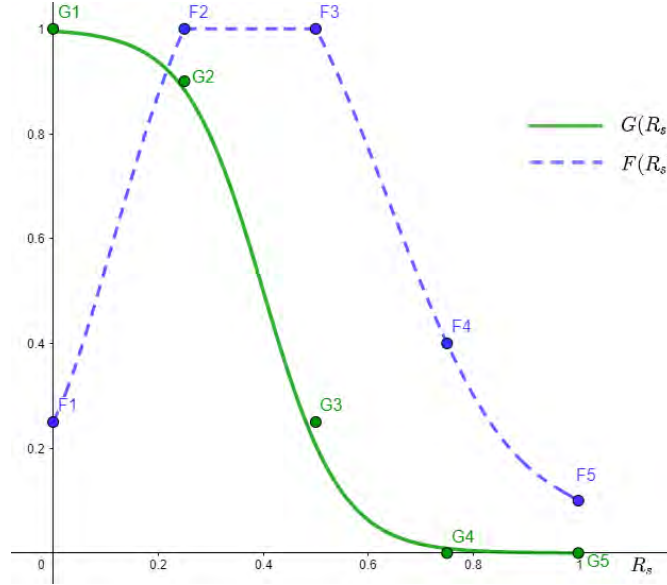


Figure 4.9: Strength functions for glints and Fresnel effect based on the roughness parameter. The outputs of $G(R_s)$ and $F(R_s)$ are multiplication factors for the glints and Fresnel effects to achieve qualitatively similar results for varying the specular roughness parameter observed during the pathtracing experiments.

Hereby, the user can set a roughness value like for the pathtracing experiments and based on this, a strength value is multiplied on the glints and Fresnel effects (the output of $G(R_s)$ and $F(R_s)$ respectively), qualitatively achieving the effects seen for roughness values 0.0, 0.25, 0.5, 0.75 and 1.0 in the pathtracing experiments. It should be noted that this does not accurately model what happens physically for different values of roughness, however only attempts to provide a qualitative approximation, an approach that has its weaknesses. For instance, it is not known what the glossy pass appears like for roughness values other than the ones used in the pathtracing experiments meaning $G(R_s)$ and $F(R_s)$ could be very incorrect for remaining values. Furthermore, qualitatively estimating the strength values as seen in Table 4.1 is prone to errors.

The final specular reflection contribution L_s can be seen in Equation 4.23. This is an addition of the Fresnel and glints contributions.

$$\begin{aligned}
 f &:= F(R_s)f \\
 g &:= G(R_s)g \\
 L_s(x, \omega_o) &= L_f(x, \omega_o) + L_g(x, \omega_o)
 \end{aligned}
 \tag{4.23}$$

Multiplying the diffuse contribution with $(1 - f)$ ensures the right relationship between specular and diffuse reflection. The diffuse contribution $L_d(x, \omega_o)$ is therefore updated as seen in Equation 4.24. It would be more accurate to also multiply the diffuse contribution by $(1 - g)$ however since the glints effect is already physically incorrect as it is multiplied by I_{glints} , it was decided to simply have the glints as an effect that is just an addition to the final contribution and not use it to regulate the diffuse contribution.

$$L_d(x, \omega_o) := L_d(x, \omega_o) (1 - f) \quad (4.24)$$

4.2.5 Transmission

The pathtracing experiments showed that for increasing transmission more light is refracted through the individual sand grains, resulting in a larger transmission contribution at macro level. It seems the transmission contribution can be divided into two parts; one that correlate with the term $n \cdot \omega_i$, i.e. where light hits the surface a transmission contribution can be seen, and one that is visible at low density areas, seemingly correlating with the term $1 - (n \cdot \omega_o)$, i.e. where the normals do not point straight at the observer. The first aspect arises from the fact that when the surface consists of many sand grains, light can travel through the grains and towards the observer, creating a uniform contribution across the surface. The latter aspect relating to low density areas is especially apparent around the edges of the sphere and the wavy plane, which to a degree can be represented by checking where the normals do not point straight at the viewer. These observations can be revisited in Figure 4.10.

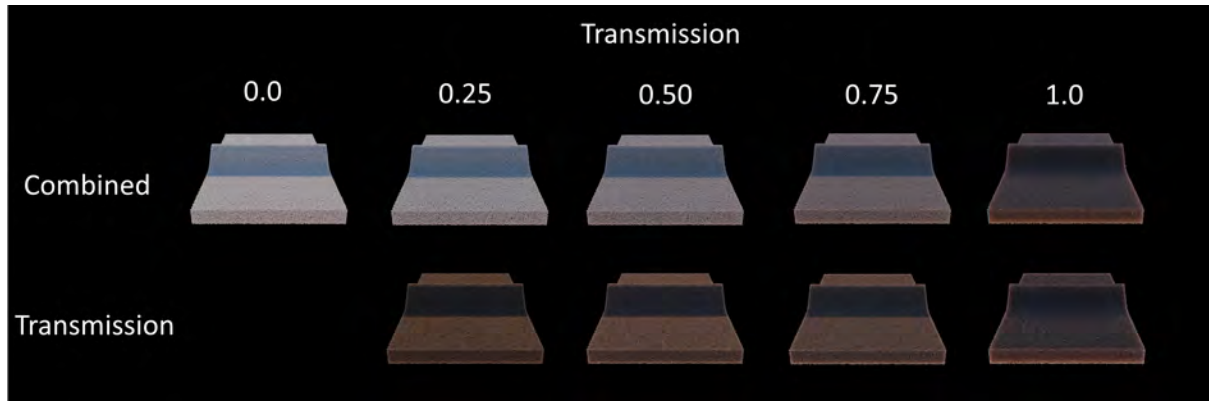


Figure 4.10: Combined and transmission lighting pass details for varying transmission values. These results lay the foundation for assessment of the transmission contribution.

The first observation related to the term $\omega_i \cdot n$ could simply be modelled using this term, similarly to the diffuse contribution. This however introduces challenges; when the light cannot be traced through grains, it is difficult to create an accurate approximation for how much of the light that arrives at the eye has been transmitted, modelling the transmission pass for the offline render results. Perhaps, instead of attempting to model the transmission pass specifically, one could model the effect that increasing transmission has on the combined result. As transmission increases, the combined result at areas correlating with high values of $\omega_i \cdot n$ becomes darker. This is most likely due to the fact that more rays are being absorbed when travelling through the collection of sand grains for increasing transmission. This could be modelled by a transmission extinction factor that can be multiplied onto the diffuse contribution L_d . When the value of transmission is 1.0, the diffuse contribution is removed in Blender, however since the methods of this project does not model the transmission pass explicitly but instead models the combined results, diffuse should not be set to 0. A minimum value should be used for the extinction factor

that will be multiplied on the diffuse contribution. Table 4.2 shows a potential function for the extinction value T_{ext} to be multiplied on the diffuse contribution. This ensures the extinction value increases with transmission but never becomes smaller than 0.1. Figure 4.10 shows that the relationship appears linear from transmission values between 0 – 0.75, but decreases drastically for value 1.0. This is represented by the values of T_{ext} in Table 4.2.

T	0.0	0.25	0.5	0.75	1.0
T_{ext}	1.0	0.85	0.7	0.55	0.1

Table 4.2: Diffuse extinction factor T_{ext} arising due to increased absorption from increasing values of T . The values in this table have been qualitatively estimated looking at the results in Figure 4.10.

The function that models the values from Table 4.2 can be seen in Equation 4.25 and the function is visualized in Figure 4.11.

$$T_{ext}(T) = \begin{cases} -3.2 T^4 + 4.8 T^3 - 2.2 T^2 - 0.3 T + 1, & \text{if } 0 \leq T \leq 1 \\ \text{not defined,} & \text{otherwise} \end{cases} \quad (4.25)$$

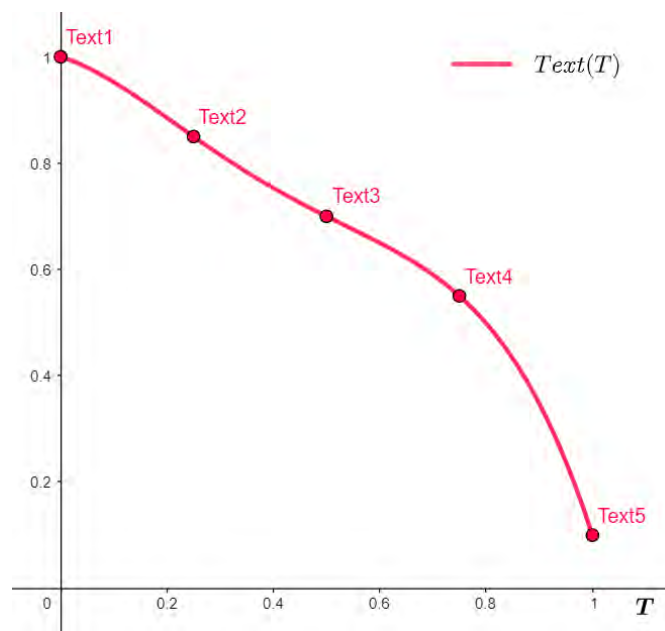


Figure 4.11: Extinction multiplication factor for L_d based on the transmission factor T . The output of $T_{ext}(T)$ will be multiplied to L_d to achieve qualitatively similar results for varying the transmission parameter observed during the pathtracing experiments.

As this function is modelled similarly to the strength functions for the specular contribution, similar limitations are present. Since only a discrete number of observations were made for the strength of the combined results based on the value of transmission, i.e. observations for values 0.0, 0.25, 0.50, 0.75 and 1.0, the values in between are unknown. Most likely, the step

drop from value 0.75 – 1.0 is wrong, however there is currently no information for how the output behaves for values in between. The diffuse contribution $L_d(x, \omega_o)$ is updated as seen in Equation 4.26.

$$L_d(x, \omega_o) := L_d(x, \omega_o) T_{ext}(T) \quad (4.26)$$

The second observed part of the transmission model that relates to light travelling through low density areas need to consider a few aspects. As described, it potentially relates to the term $1 - (n \cdot \omega_o)$, i.e. where the normals do not point straight at the viewer, however in comparison to the other part of the transmission model, this contribution is only observable when the light is on opposing sides of the observer in relation to the surface point where light from the main light source can travel more easily through the grains to the observer. The strength of this effect seems to increase linearly with the value of transmission T . These considerations can be seen formalized in Equation 4.27 as t .

$$t = T (1 - \max(0, n \cdot \omega_o)) \max(0, -\omega_o \cdot \omega_i) \quad (4.27)$$

where the left-most part of the equation excluding T represents areas where the macro normal is not pointing straight at or away from the viewer, and the right most part represents cases where the negated view direction $-\omega_o$ aligns with the light direction ω_i , i.e. the light is on opposing sides of the surface point in relation to the observer. Including the multiplication by T ensures the strength of the expression increases for increasing values of transmission T .

Another variable that relates to the transmission expression is the transmission roughness. The results of this can be revisited in Figure 4.12.

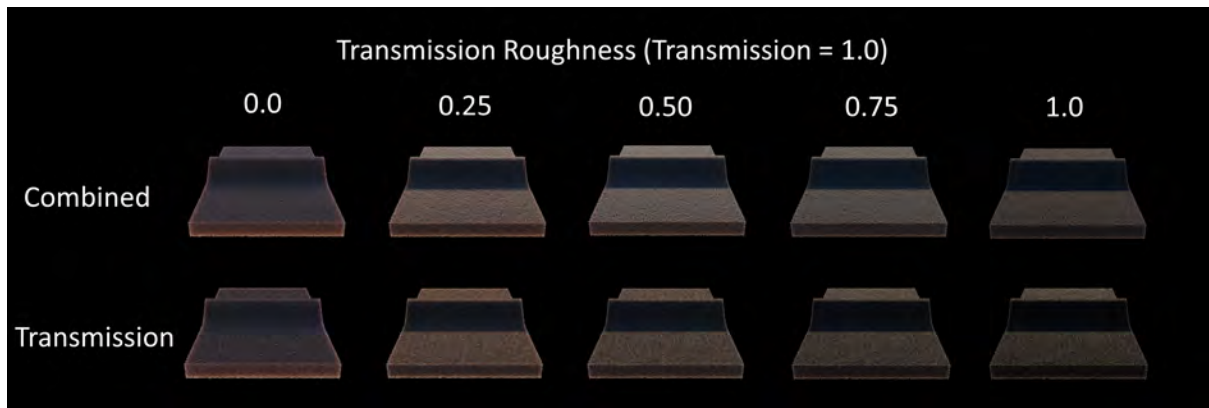


Figure 4.12: Combined and transmission lighting pass results for varying transmission roughness values. These results lay the foundation for assessment of the transmission contribution.

To model the roughness of transmission, the lobe of the transmission can be controlled. One could take inspiration from the specular part of the Phong reflection model in which the reflection lobe is controlled by shininess value [26]. Here, the specular component is raised to the power

of the shininess value. As the shininess is inversely correlated with roughness, the transmission contribution should be raised to $1/roughness$. This inspired updating the expression for t to model roughness. This can be seen in Equation 4.28.

$$t = T (1 - \max(0, n \cdot \omega_o))^{1/R_t} \max(0, -\omega_o \cdot \omega_i) \quad (4.28)$$

where t is the factor that determines the amount of transmitted light, T controls the strength of transmission, and R_t is the transmission roughness factor. Large values of R_t means a larger lobe of transmitted light. For instance, observing an edge of a sphere, the light will *bleed out* from the edge, affecting more points on the sphere. Experiments showed that the most accurate results were obtained when scaling R_t between $[0.2, 0.4]$. To make the user parameters intuitive, the roughness value R_t should still be in range $[0, 1]$ in the user interface, however should be scaled when used in the transmission calculation in Equation 4.28. The scaling operation can be seen in Equation 4.29.

$$R_t^* = 0.2 + R_t (0.4 - 0.2) \quad (4.29)$$

Furthermore, it can be observed that as roughness increases, the transmission contribution loses strength almost linearly with increasing roughness. The strength for transmission roughness value 0.0 is hard to interpret in relation to the rest, however multiplying it by 1 makes most sense as it otherwise would change the results for the established output for transmission value 1.0. A strength function for transmission roughness can be introduced as it was for glints and Fresnel. Qualitatively, it seems the strength is minimum 0.1 for transmission roughness value 1.0. A function $R_t^{strength}$ is introduced that interpolates linearly between 1.0 and 0.1 for increasing values of R_t . This will be multiplied on the final transmission contribution. The formula of the function for calculating $R_t^{strength}$ can be seen in Equation 4.30 and inspected visually in Figure 4.13.

$$R_t^{strength}(R_t) = \begin{cases} 1 - 0.9 R_t, & \text{if } 0 \leq R_t \leq 1 \\ \text{not defined,} & \text{otherwise} \end{cases} \quad (4.30)$$

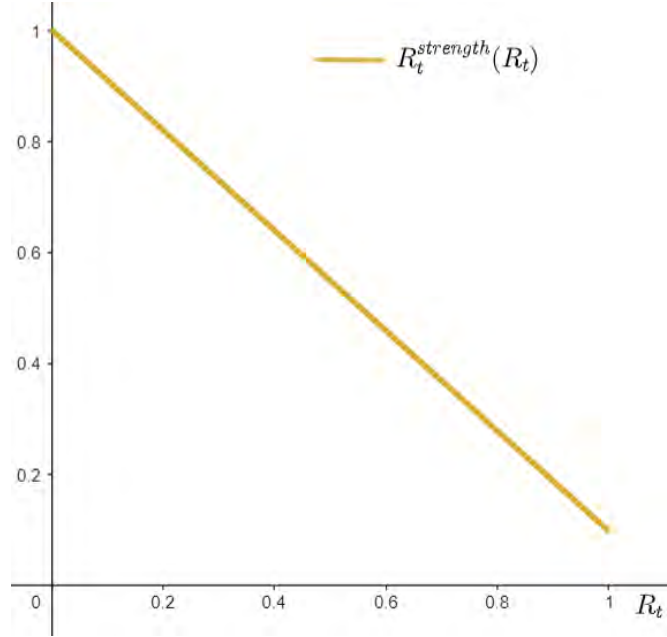


Figure 4.13: Strength of transmission contribution $R_t^{strength}$ based on transmission roughness R_t . The output of $R_t^{strength}$ will be multiplied to t to achieve qualitatively similar results for varying the transmission roughness parameter observed during the pathtracing experiments.

The final calculation of t can be seen in Equation 4.31, which includes all elements that have been described in this section.

$$t = R_t^{strength} T (1 - \max(0, n \cdot \omega_o))^{\frac{1}{R_t^*}} \max(0, -\omega_o \cdot \omega_i) \quad (4.31)$$

The refracted light should be shaded by the incoming light from the main light source, and since it is refracted through the sand grains, it should also be affected by the color of the sand grains. Since the transmission contribution here models the transmission of light through multiple sand grains, the average sand grain color can be used. Since the skylight is sampled at x , it does not represent the skylight that has been transmitted through the sand, and therefore it should not be used. This is a limitation, since it is not just light from the main light source that is transmitted through. This description is formalized in Equation 4.32.

$$L_t(x, \omega_o) = \rho_n L_i(x, \omega_i) t \quad (4.32)$$

where $L_t(x, \omega_o)$ is the final transmission contribution, ρ_n is the macro color of the sand grains, $L_i(x, \omega_i)$ is the color of the incoming main light source, and t is the amount of transmitted light. The reason why ρ_n is used and not ρ_m is that there is no measure of how many sand grains the light has travelled through, and therefore an average color is more correct to use than the color of the sand grain that the light exits from before reaching the eye.

Usually, the diffuse contribution would likewise be controlled by how much light is transmitted, however the diffuse contribution $L_d(x, \omega_o)$ models the light arriving at the point where the transmissive contribution $L_t(x, \omega_o)$ models light coming from a different light path, representing light that has travelled through the collection of grains. Therefore, the transmissive contribution will not be used to control the amount of diffuse contribution for the point. T_{ext} is however multiplied on the diffuse contribution to model the extinction of light due to increasing transmission.

A limitation to the process that has been described in this section is the lack of knowledge about results for varying transmission roughness for other values of transmission than 1.0. Varying transmission roughness for other values of transmission could provide very different insights, something that was not considered in the presented methods. The presented transmission method approximates the transmission contribution that can be observed through sand grains at low density areas for cases where the light source is on opposing sides of the geometry in relation to the observer. It does however not directly model low density areas. For instance, a uniformly thin plane of sand grains illuminated by a light source on opposing sides of the plane in relation to the camera should have an equally intense transmission contribution. Using the method in this project, the plane would not be affected by the transmission contribution at all, as it does not directly model the density of geometry a light ray has passed through before reaching the point to be shaded, which is a limitation. An observation was presented in this section that correlated the transmission contribution and the term $\omega_i \cdot n$, where light rays could be transmitted through sand grains across the surface to the observer. This observation was not directly modelled as part of the transmission contribution, which is a limitation. More experiments could be made to offer a more qualified guess as to how this effect can be achieved for a real-time solution as accurately as possible. The effect of this on the combined result was however approximated by introducing a diffuse extinction factor. Therefore, while the transmissive contribution of the methods of this project significantly differs from the results of the transmissive pass in the pathtracing results, it is hoped that the combined results will approach the observations from the pathtracing results.

4.3 Transitioning from Micro to Macro Scale Shading

The previous section outlines the shading components of this project, however does not confront the issue identified in Section 2.3 regarding the transition effect observed for varying distances to the observer. If individual sand grains visually differ, single sampling will not be adequate for greater distances as more grains should contribute to a single pixel. The methods of this project introduces granularity and therefore visual differences between individual grains, and single sampling will be used to abide to real-time constrains. This means a maximum of one sand grain will contribute to a pixel at greater distances. A method for dealing with this must be derived so that the necessary averaging affect at greater distances is modelled.

This issue was examined by constructing a shader that utilizes a diffuse lookup in an HDR image using micro normals, derived using the approach from Section 4.1. This was rendered using

single sampling and multi sampling, the latter using 100 samples. Multi sampling was done using the derivative of the current fragment position in world space in respect to the screen-space coordinates, and multiplying this with a random value in a predefined set of random values, essentially *jittering* the sample position. A plane was rendered using this shader, and the results can be seen in Figure 4.14.

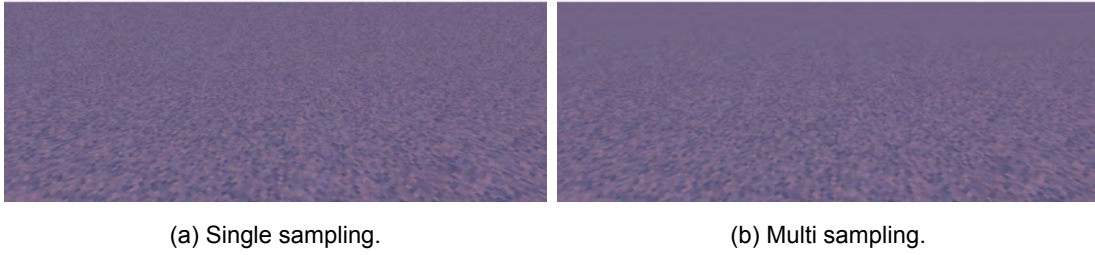


Figure 4.14: Rendering results for single sampling (left) and multi sampling with 100 samples per pixel (right). The shader uses a diffuse lookup in an HDR image using micro normals. For multisampling, the sampling positions were found using the derivative of the current fragment position in respect to the screen-space coordinates, multiplied by a random value.

These results show that using multi sampling, the results of pixels at greater distances become averaged, whereas for single sampling the granularity is still observable at far distances. As described, multisampling is too expensive for a complex shader for real-time usage. Therefore, this cannot be implemented directly, but instead two things must be approximated; the shading of the sand at greater distance that models the averaging effect observed, and a function that determines how to transition between the shading outlined in the previous section and the shading modelling the effect observed for multisampling at greater distances. This section deals with how to shade the sand at greater distances and how to transition between the two approaches. The approach presented in previous sections will be referred to as L_{close} , and the one presented in this section as L_{far} . L_{close} is a combination of the contributions presented in previous sections and can be seen in Equation 4.33.

$$\begin{aligned}
 L_{close}(x, \omega_o) &= L_d(x, \omega_o) + L_s(x, \omega_o) + L_t(x, \omega_o) \\
 L_{close}(x, \omega_o) &:= L_{close}(x, \omega_o) rhom^{intensity}
 \end{aligned}
 \tag{4.33}$$

where $L_d(x, \omega_o)$ is the diffuse reflection towards the observer at surface point x , $L_s(x, \omega_o)$ is the specular contribution including Fresnel and glints, and $L_t(x, \omega_o)$ is the transmissive contribution. The entire contribution is multiplied by the intensity value of the sand grain $rhom^{intensity}$.

The current components that depend on the micro normal, and therefore are not directly usable for shading at a distance, are sand grain colors, intensity and glints. The appearance study and field trip showed that at a greater distance, glints are no longer observable. This is reasonable, as they are averaged out by the contribution of neighboring sand grains. Furthermore, while

the sand in photos still appear diffuse at a distance, the granularity of the sand is no longer observable. In this project, the diffuse granularity of the sand stems from the sand grain colors and intensity values. An idea could be to simply utilize the same color rho_n and intensity $rho_n^{intensity}$ for all sand grains at a distance and use the same diffuse shading outlined in previous sections. However, this does not accurately capture the differences at micro level. Instead, a micro facet reflection BRDF could be used. Playdead used Oren-Nayar [5], and while this was suitable under most conditions, it fell short under some. Specifically, when the main light source was opposite the observer in relation to the sand. Possibly, this is caused by the fact that Oren-Nayar on its own does not account for the specularity and transmissive properties of sand. Utilizing Oren-Nayar in conjunction with Fresnel and the custom transmission method could resolve the issues observed. As a result of these considerations, it was decided to utilize Oren-Nayar for diffuse shading at greater distances combined with Fresnel specular reflectance and transmission. The Oren-Nayar BRDF models the micro structure as having small Lambertian *V-cavities*, which in the case of sand would represent sand grain facets at micro level. By having a low variance in facet orientation angles, this could approach the cube micro normals used for the methods of this project. Variance is introduced as σ in the Oren-Nayar formula, as can be seen in Equation 4.34 [27].

$$\begin{aligned}
L_{oren}(x, \omega_o) &= A + B \max(0, \cos(\phi_i - \phi_o)) \sin\alpha \tan\beta \\
A &= 1 - \frac{\sigma^2}{2(\sigma^2 + 0.33)} \\
B &= \frac{0.45\sigma^2}{\sigma^2 + 0.09} \\
\alpha &= \max(\theta_i, \theta_o) \\
\beta &= \min(\theta_i, \theta_o)
\end{aligned} \tag{4.34}$$

where θ_o and θ_i are the elevation angles between the view direction and the surface and the light direction and the surface respectively, and ϕ_o and ϕ_i represents the azimuth angles. σ is used to model the variance of the orientation angles of the micro structure V-cavities and hereby the roughness of the surface. For $\sigma = 0$, Oren-Nayar simplifies to the Lambertian BRDF model. More experiments could have been made to test which BRDF would be the most optimal to use to approximate the averaging effect seen for granular materials at a distance, and Oren-Nayar is not necessarily the optimal one to use. As Playdead had experienced, it was experienced that Oren-Nayar appeared very dark when camera pointed toward the main light source. It was solved by using very low σ values for Oren-Nayar, however this means it approaches Lambertian shading, which is not optimal. For this project, it will be used to solve the challenges presented, however future experiments should be carried out to investigate if other methods or BRDFs are more suitable.

To model subsurface scattering, the same approach as presented previously will be used. Oren-Nayar is multiplied by the term $\max(0, n \cdot \omega_i)$ to ensure the output depends on how illuminated the surface is by the light. Like for the other diffuse approach, the light from the main light source $L_i(x, \omega_i)$ and environment light $L_{sky}^\Omega(x, \Omega)$ should be used. The diffuse color

used for Oren-Nayar should be an average of the set of diffuse colors of the sand grains, i.e. ρ_n . $L_{brdf}(x, \omega_o)$ represents the final diffuse contribution at far distances. Just as for the diffuse contribution at close distances, $L_{brdf}(x, \omega_o)$ should be multiplied by $(1 - f)$ and the transmission extinction factor T_{ext} . This is summarized in Equation 4.35, that shows the entire calculation of the diffuse contribution at far scales $L_{brdf}(x, \omega_o)$.

$$\begin{aligned}
L_{brdf}(x, \omega_o) &= \frac{\rho_n}{\pi} L_i(x, \omega_i) (f_s + (1 - f_s) L_{oren}(x, \omega_o) \max(0, n \cdot \omega_i)) \\
L_{brdf}(x, \omega_o) &:= L_{brdf}(x, \omega_o) + \frac{\rho_n}{\pi} L_{sky}^\Omega(x, \Omega) \\
L_{brdf}(x, \omega_o) &:= f_e L_{brdf}(x, \omega_o) \\
L_{brdf}(x, \omega_o) &:= (1 - f) L_{brdf}(x, \omega_o) \\
L_{brdf}(x, \omega_o) &:= T_{ext} L_{brdf}(x, \omega_o)
\end{aligned} \tag{4.35}$$

Just as the diffuse contribution at distances should model the irregular micro surface structure arising from sand grains, so should the specular and transmissive contributions ideally. One could for instance use specular micro-facet BRDF models. While this would be more correct, this was not managed within the scope of this project. Instead, the same specular and transmission contributions that were presented for L_{close} that depend on the macro normal n were used, except for $L_g(x, \omega_o)$, as glints fade at greater distances. The final shading for far distances $L_{far}(x, \omega_o)$ can be seen in Equation 4.36.

$$\begin{aligned}
L_{far}(x, \omega_o) &= L_{brdf}(x, \omega_o) + L_f(x, \omega_o) + L_t(x, \omega_o) \\
L_{far}(x, \omega_o) &:= L_{far} rhO_n^{intensity}
\end{aligned} \tag{4.36}$$

Like for the shading at close distances, and the entire contribution is an addition of the diffuse, specular, and transmissive contribution. Notice, that compared to shading at close distances, only the Fresnel contribution $L_f(x, \omega_o)$ is used for the specular contribution. The entire contribution is multiplied by the the average intensity values of sand $rhO_n^{intensity}$.

Next, a function should be made that can model the transition from the shading of sand at closer distances to the one at greater distances. A simple idea could be to interpolate between the $L_{close}(x, \omega_o)$ and $L_{far}(x, \omega_o)$ based on the distance from the point x to the observer. A function that can be used that smoothly interpolate between two values is the sigmoid function [25]. This function outputs values in the range $]0, 1[$ and approaches 0 as x approaches $-\infty$ and approaches 1 as x approaches ∞ . The sigmoid function can be seen in Equation 4.37.

$$sig(x) = \frac{1}{1 + e^{-a(x-b)}} \tag{4.37}$$

where a is a value that can be used to *stretch* the function in the x -direction, and b is used to shift the function in the x -direction. Examples of this are illustrated in Figure 4.15 for values of $a = 10$ and $a = 2$ and for values of $b = 1$ and $b = 2$.

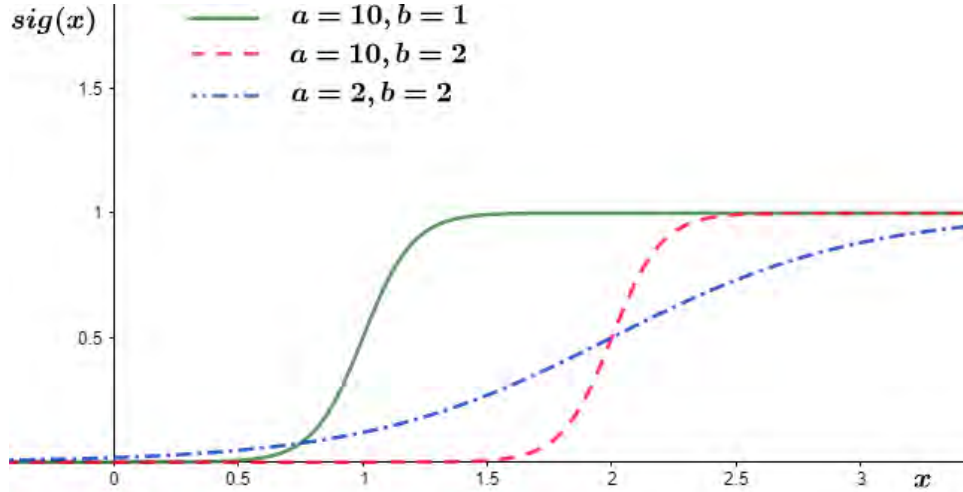


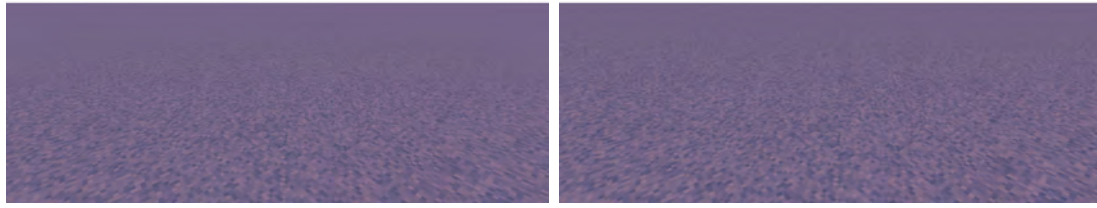
Figure 4.15: Sigmoid functions for $a = 10, b = 1$ (green solid line), $a = 10, b = 2$ (red, striped line) and $a = 2, b = 2$ (blue striped and dotted line). As a decreases, the function is stretched across the x -axis, and when b increases the function is shifted in a positive direction on the x -axis.

$sig(x)$ can be used to interpolate between the shading at close distance at at a far distance using Equation 4.38.

$$L_{final}(x, \omega_o) = (1 - sig(dist)) L_{close}(x, \omega_o) + sig(dist) L_{far}(x, \omega_o) \quad (4.38)$$

where $L_{final}(x, \omega_o)$ is the final shading at surface point x in direction ω_o toward the viewer, $dist$ is the distance between x and the world space camera position, and $L_{close}(x, \omega_o)$ is the shading function for shading at close distances and $L_{far}(x, \omega_o)$ is the shading function for shading at far distances. Values for a and b in the sigmoid function naturally depend on the size of sand grains, i.e. larger sand grains mean that larger distances are required before single sampling is no longer appropriate.

Utilizing this function to change between single sampling using micro normals and single sampling using the macro normal yields the results seen in Figure 4.16a. Here, single sampling using micro normals represents shading closer to the camera, and single sampling using macro normals represents shading further from the camera where averaging results were seen for multisampling. Values $a = 0.3$ and $b = 10$ have been used. Figure 4.16b shows the results where multi sampling with 100 samples is used entirely. It can be seen, the two appear very similar, suggesting this is an appropriate function for the purpose of transitioning between shading at close scale and far. Figure 4.17 illustrates the function used in Figure 4.16a however interpolating between red and blue instead of results for micro and macro normal diffuse shading, allowing to more easily inspect the function visually.



(a) Interpolation between using micro and macro normals for skybox lookup.

(b) Multi sampling.

Figure 4.16: **Left:** the result of using $sig(x)$ to interpolate between shading using micro normals and using macro normals, where x is the distance from the point being shaded to the observer, $a = 0.3$ and $b = 10$. **Right:** shading using micro normals and utilizing multisampling with 100 samples.



Figure 4.17: Illustration of function used in Figure 4.16a where interpolation is made between red and blue instead of between shading using micro normals and macro normals.

The approach presented in this section provides one way to approximate the effect of multi sampling that is necessary to approach realistic results. However, it is not exhaustive for representing all scales of sand. It does provide a smooth transition between two scales as was observed during the field trip and photos of sand, but does not consider distances smaller than where $L_{close}(x, \omega_o)$ is accurate or distances greater than where $L_{far}(x, \omega_o)$ is accurate. As described, the specular and transmissive contributions at far distances at current time do not consider the micro structure of the surface arising from the sand grains, which is a limitation. Finally, it should be investigated whether other methods are more appropriate for representing sand at far distances, potentially BRDF methods as the one presented by D'eon [4].

4.3.1 Shading Overview

The diagrams in this section show the overview of the shading method presented in the previous sections. This will lay the foundation for the implementation.

Figure 4.18 shows the parameters that will be exposed to the user and which shading contribution they influence as well as an overview over how the final shade is derived for this project. There are six user-controlled parameters that were chosen inspired by the exposed parameters in Blender's principled BSDF; P controlling porosity, $C[8]$ the set of 8 sand grain colors,

SSS controlling the amount of subsurface scattering, the specular roughness value R_s , the transmission parameter T and the transmission roughness parameter R_t . It was decided to expose the porosity value, as this allows the user to specify the porosity of the sand grains which essentially correspond to the distance between distributed sand grains for an offline rendering approach rendering explicit sand geometry. The calculations of L_d (diffuse reflection), L_s (specular reflection), L_t (transmission), L_{brdf} (Oren-Nayar BRDF) and L_f (Fresnel reflection, i.e. L_s excluding glints) has been presented throughout the previous sections. The diffuse reflectance is affected by user parameters C [8] and SSS , the specular reflectance is affected by R_s and the transmission is T and R_t as well as C [8]. All shading contributions are affected by the porosity value P . L_{close} is the final shading at close distance to the observer, representing where single sampling can be used to approximate accurate results, and L_{far} is the shading at greater distances where a BRDF is used to approximate the results of multisampling. L_{final} is the final shade that is controlled using the sigmoid function from Equation 4.37 (represented by a sigmoid symbol in the figure) and the distance from the point to the observer. The symbols \oplus represent addition of inputs.

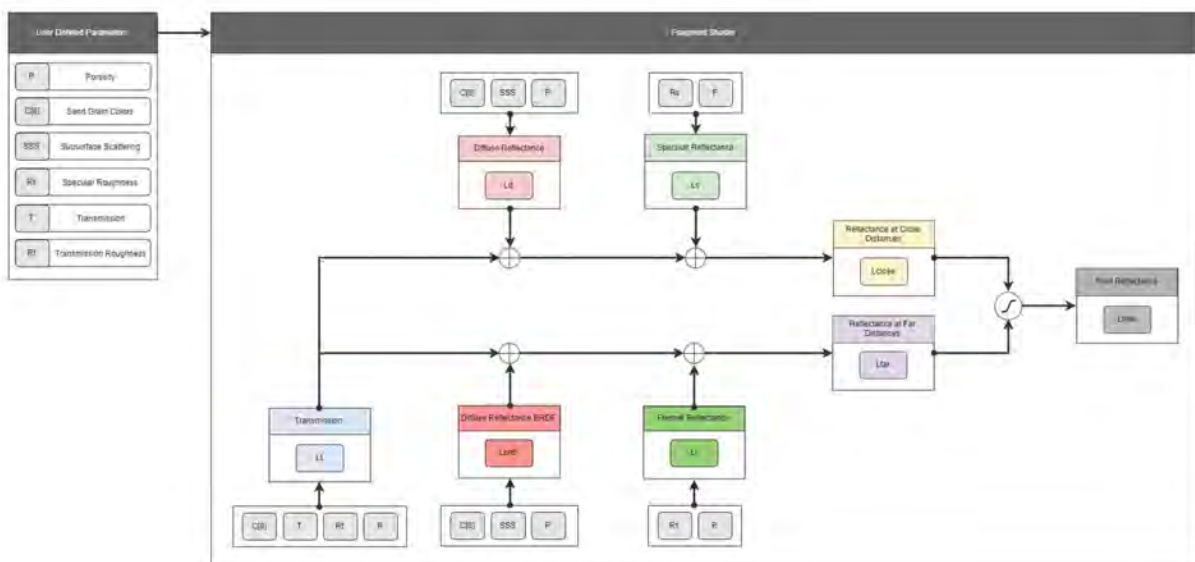


Figure 4.18: Overview of user defined parameters and which parts of the shading contribution they influence, and how the final shade is derived using the methods of this project.

4.4 Method Conclusion

The methods presented throughout this section have attempted to model the appearance observations made during the appearance study, using the appearance study of sand and the offline pathtracing results as a frame of reference. A list of challenges that should be considered for a real-time solution for rendering sand at multiple scales was devised and presented in Section 2.3, based on the analysis. The methods of this project attempted to deal with the presented challenges, and methods for dealing with most has been devised throughout.

Limitations to the methods and how they have been derived have been described throughout the section for each of the approaches. Results for the implementation of the methods will be presented and evaluated in later sections, and the degree to which the method is able to approximate the findings of the analysis and deal with the listed real-time challenges will be discussed.

5 Implementation

The implementation of this project was done in Unity (version 2021.1.11f1) [28], using its Universal Render Pipeline (URP) that uses forward rendering. Unity was chosen since it is a game engine and hereby the implementation can be tested in environments as close to the intended use as possible. Furthermore, Unity supports custom shaders written in CG and HLSL. This project uses CG. Despite being implemented in CG in Unity, the implementation and methods should be generalizeable and easy to translate to use for other applications. A shader was created in CG that passes object information through a vertex shader to the fragment shader, and all calculations described in the methods were done in the fragment shader. The entire shader can be inspected in Appendix A.

Issues were experienced working with Unity URP for the implementation. URP has a very different shader workflow than its built-in render pipeline, and documentation on custom shaders in URP is very limited. When building scenes using custom shaders, the results differed greatly from the ones observed in its internal game mode, making the process of building scenes difficult. One of the issues identified with this process was the fact that after building the scene, the shader lost connection to the directional light in the scene, rendering the material black, meaning light direction and color had to be defined explicitly in the shader for builds. The solution to this was not found. The shader is furthermore not always able to return the environment light directly, even when multiplied by a great factor. This will render the result black. However, the same environment lighting information is clearly visible in the final result when used in the reflectance calculations. The reason for this is not understood, and this made the development process difficult, especially for the multi sample experiments that directly utilizes the environment lighting. Furthermore, temporal anti-aliasing using motion vectors is not supported currently for URP, which is a limitation for this project. The explicit approach for sampling environment light presented in the methods of this project was not used for the implementation. Unity has functions for sampling the environment light from an HDR skybox. It can sample different levels of LODs of the skybox and hereby model light that is reflected diffusely and specularly, and these functions were used for this project instead. Finally, implementing shadows in custom shaders in Unity URP proved difficult and was not managed within the scope of this project, affecting the results.

6 Results

The results presented in this section were created using a laptop computer with an 11th Gen Intel(R) Core(TM) i7-11800H 2.30 GHz processor, 16 GB RAM and an NVIDIA GeForce RTX 3080 Laptop GPU.

This section will present results for comparison with the offline pathtracing render results, render times, a breakdown of the shading contributions of the real-time shader, and builds to allow to experience the implementation in real-time. The presented render results and their assessment will be qualitative, as it has not been possible to acquire ideal render results that can be quantitatively compared to the results of this project.

6.1 Parameter Comparison

The following figures show comparison between pathtracing results in Blender and real-time rendering results from Unity for varying values of the user specified parameters subsurface scattering, specular roughness, transmission, and transmission roughness. The results are shown for a sphere and wavy plane. The results from Blender are the same as presented during the appearance study. The combined render results are shown as well as the lighting passes affected by the parameter in question.

Unfortunately, there is a noticeable color difference in the renders from Blender and Unity. This affects both the HDRI and the material and was not tackled during generation of the results. It is believed it is due to the two programs potentially utilizing different color spaces. The colors should not be directly compared, but instead the relevant aspects relating to the development of intensity differences across the results should be considered. Furthermore, shadows as described were not properly implemented, further affecting the results.

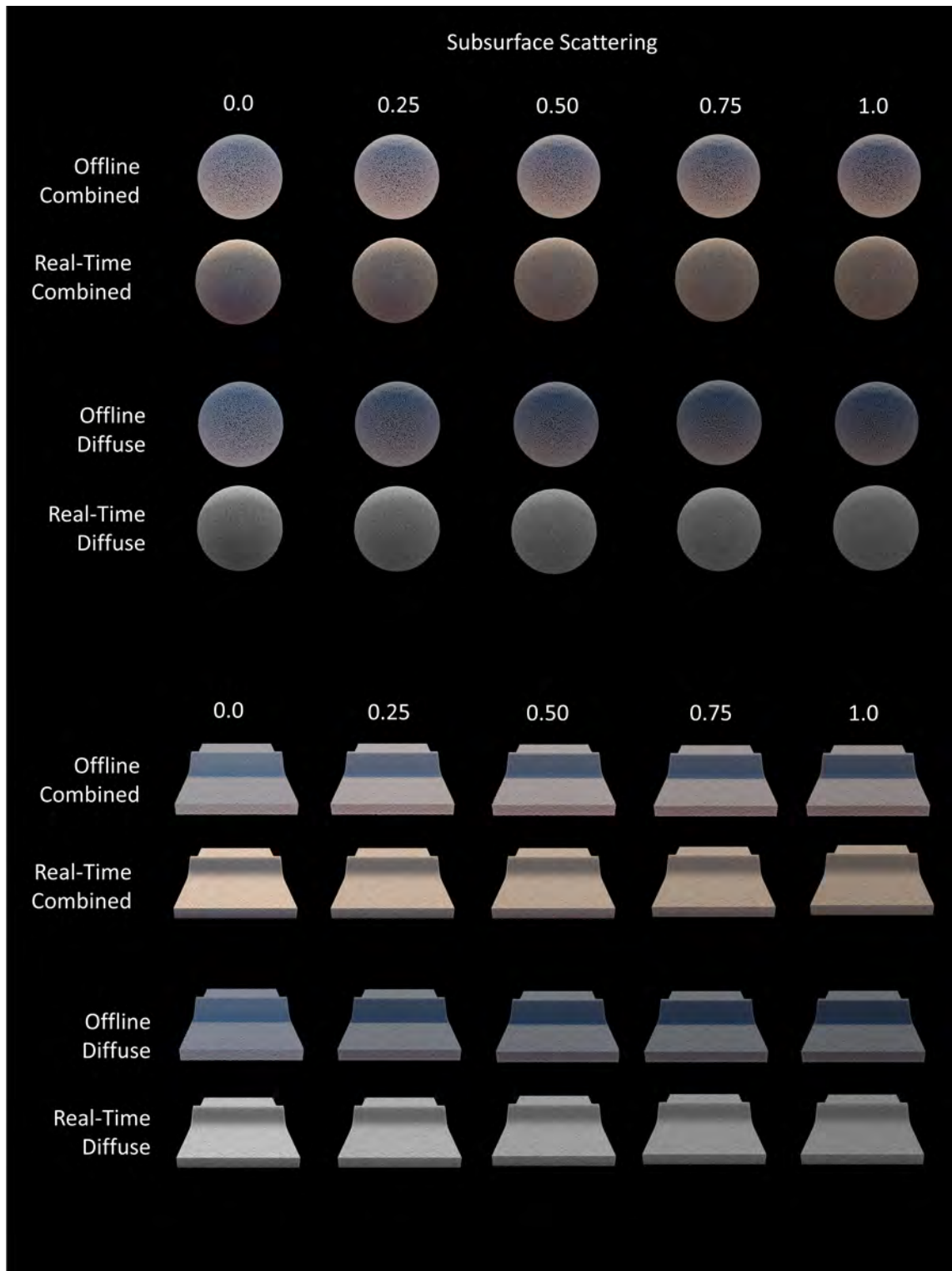


Figure 6.1: Results for changing the subsurface scattering parameter for the BSDF in Blender (offline) and the shader in Unity (real-time) for values of 0.0, 0.25, 0.50, 0.75 and 1.0. **Row 1 and 2 (from top):** combined render results for sphere of sand for Blender and Unity respectively. **Row 3 and 4 (from top):** render results for sphere of sand for diffuse pass for Blender and Unity respectively. **Row 5 and 6 (from top):** combined render results for wavy plane of sand for Blender and Unity respectively. **Row 7 and 8 (from top):** render results for wavy plane for diffuse pass for Blender and Unity respectively.

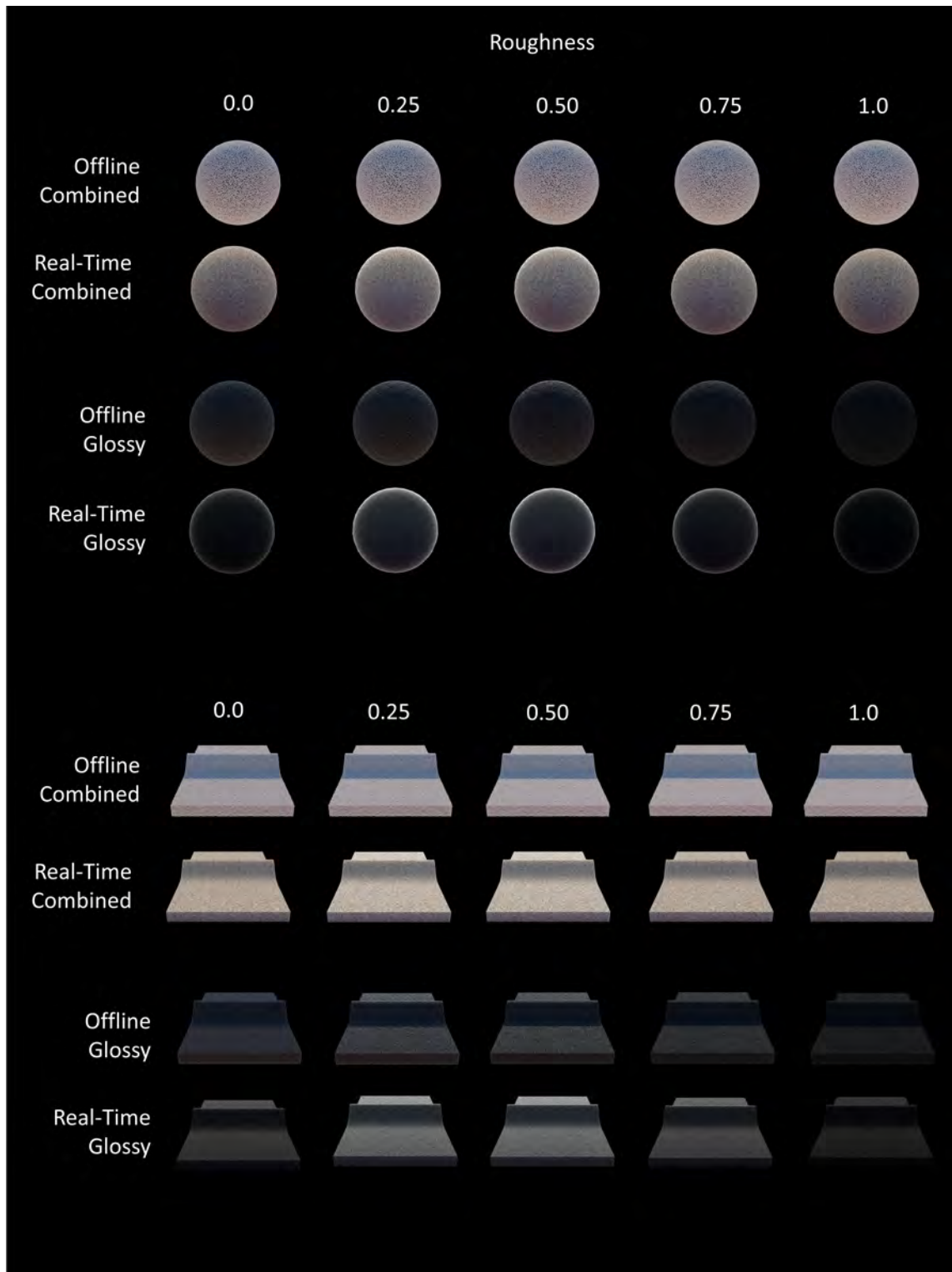


Figure 6.2: Results for changing the roughness parameter for the BSDF in Blender (offline) and the shader in Unity (real-time) for values of 0.0, 0.25, 0.50, 0.75 and 1.0. **Row 1 and 2 (from top)**: combined render results for sphere of sand for Blender and Unity respectively. **Row 3 and 4 (from top)**: render results for sphere of sand for glossy/specular pass for Blender and Unity respectively. **Row 5 and 6 (from top)**: combined render results for wavy plane of sand for Blender and Unity respectively. **Row 7 and 8 (from top)**: render results for wavy plane for glossy/specular pass for Blender and Unity respectively.

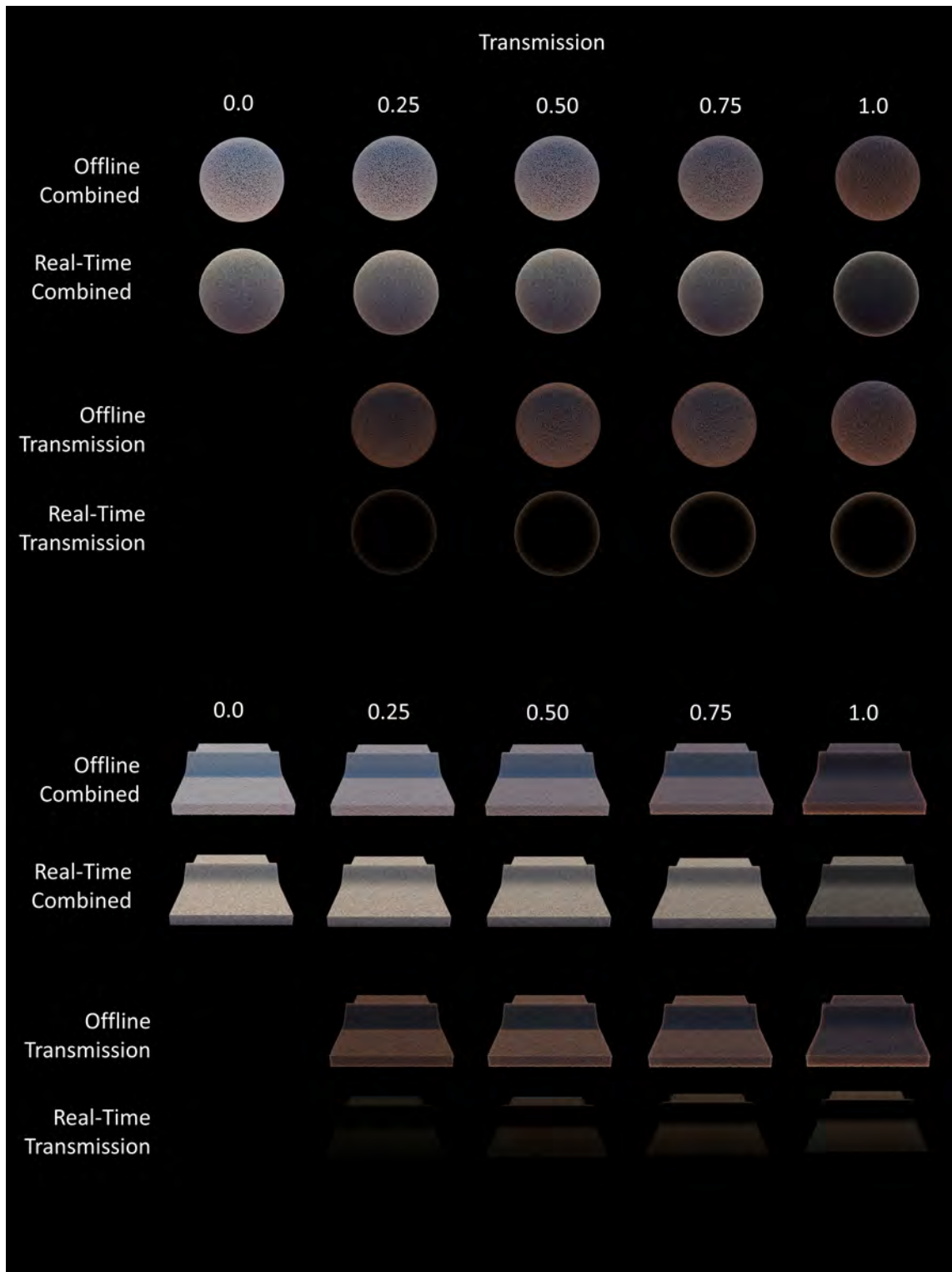


Figure 6.3: Results for changing the transmission parameter for the BSDF in Blender (offline) and the shader in Unity (real-time) for values of 0.0, 0.25, 0.50, 0.75 and 1.0. **Row 1 and 2 (from top)**: combined render results for sphere of sand for Blender and Unity respectively. **Row 3 and 4 (from top)**: render results for sphere of sand for transmissive pass for Blender and Unity respectively. **Row 5 and 6 (from top)**: combined render results for wavy plane of sand for Blender and Unity respectively. **Row 7 and 8 (from top)**: render results for wavy plane for transmissive pass for Blender and Unity respectively.

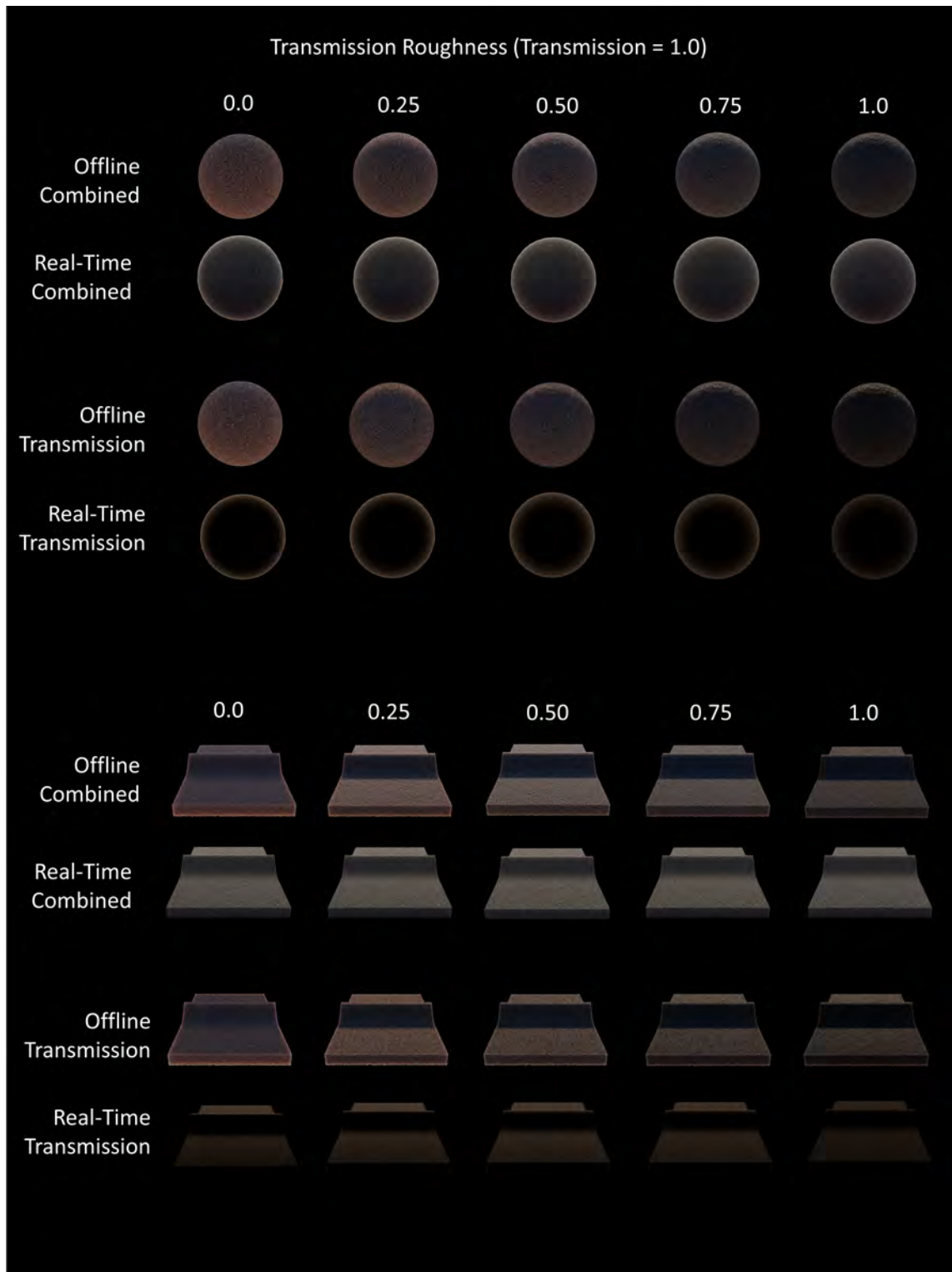


Figure 6.4: Results for changing the transmission parameter for the BSDF in Blender (offline) and the shader in Unity (real-time) for values of 0.0, 0.25, 0.50, 0.75 and 1.0. **Row 1 and 2 (from top)**: combined render results for sphere of sand for Blender and Unity respectively. **Row 3 and 4 (from top)**: render results for sphere of sand for transmissive pass for Blender and Unity respectively. **Row 5 and 6 (from top)**: combined render results for wavy plane of sand for Blender and Unity respectively. **Row 7 and 8 (from top)**: render results for wavy plane for transmissive pass for Blender and Unity respectively.

6.1.1 Discussion

As described, the color difference between Unity and Blender affects the results of the renders presented in this section, as well as the lack of shadows. Looking aside from these limitations, the following will discuss each of the presented parameters in terms of how well they approximate the offline results.

Looking at the results for subsurface scattering (Figure 6.1), the results capture the decrease in intensity due to higher absorption from subsurface scattering, meaning as the value of subsurface scattering increases, the result becomes darker. The real-time implementation in which the shading becomes less normal-dependent attempted to model that more light from the environment exits at each point for increasing subsurface scattering. It seems Unity reads the HDRI differently than Blender does; this can be seen in the Figure for subsurface scattering value 0.0, where in Blender, the sphere is lighter in the bottom (reflections from beach) than the top (reflections from the blue sky) and the opposite is the case or Unity. This could be due to the fact that the implementation in Unity does not treat intensities stored in the HDRI, and the sand is likely reflecting higher intensities than the sky. Nonetheless, the subsurface scattering present an approximation that is modelled with what physically takes place in mind.

Looking at the glossy results for varying specular roughness (Figure 2.15), overall the specular reflectance quite closely models the glossy pass from the offline results. However, the contribution seems too bright for roughness values 0.25 – 0.5. A simple solution could be to adjust the maximum strength output of the functions $F(R_s)$ and $G(R_s)$. It furthermore seems that while the glints contribution of the real-time model closely models the offline for roughness value 0, it seems more glints should be present for roughness 0.25 – 0.5 when comparing these. As for subsurface scattering, it seems to be an issue that the Unity implementation doesn't use the intensities stored in the HDRI, and the real-time implementation does not properly represent reflections from the beach sand at the bottom of the sphere and reflections from the sky at the top of the sphere.

Looking at the results, the contribution that is most dissimilar relates to transmission, which can be seen in Figure 6.3 and 6.4. The reason this contribution is so difficult to accurately approximate is the fact that it requires looking at light paths that are not accessible when shading the mesh using vertex and fragment shaders. This is especially apparent for the front of the sphere and across the top of the surface of the plane and at its edges. Since the calculations furthermore do not utilize environment information, much of the contribution is lost, for instance at the edges in the bottom of the wavy plane. The intensity development in the combined results for varying transmission values (Figure 6.3) suggest introducing the diffuse extinction factor due to increasing absorption for increasing transmission was appropriate, as it approximates the decreasing intensity of the offline combined results. The results for varying transmission roughness (Figure 6.4) do seem to partially approximate the offline results when considering the development of intensities and the specific areas of aggregate objects it attempts to model. For instance, at low density areas for the offline results (at the edges of objects) the light is

being transmitted less concentrated for increasing roughness, which is approximated looking at the edges of the sphere of the real-time results. Overall, while the approach partially approximates some of the observations for varying transmission values it fails to capture many aspects. Further investigating how to approximate the results without the knowledge of the explicit lights paths could be conducted.

6.2 Pathtracing Comparison for HDRI Matching

To evaluate the implementation of this project, it is relevant to compare it to what can be achieved using physically accurate methods, such as pathtracing. This section presents results from visually matching digital sand grains to an HDRI background in Blender. The results of this will be compared to the real-time shader in Unity, using the same user parameter values as used in Blender to investigate the similarity of the results and how close either approach can approximate the appearance of real sand.

The scene used in Section 2.2 for the pathtracing experiments was set up to create a wavy sand plane geometry that visually approximated the sand in the HDR image. Values for the BSDF were chosen to attempt to match the sand in the image. This was done in a qualitative manner in which parameters were tuned until it was believed that the sand looked convincing from as many angles as possible. Figure 6.5 shows the chosen BSDF, and Figure 6.6 shows renderings of a single grain using the BSDF.

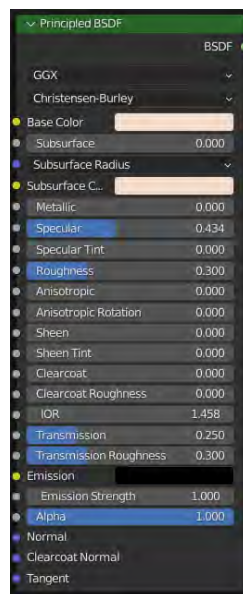


Figure 6.5: BSDF values chosen for sand grains in Blender to match the sand in the background of the HDRI. The values for the BSDF were set to $Specular = 0.434$, $Roughness = 0.300$, $IOR = 1.458$ (IOR of silica), $Transmission = 0.250$, $TransmissionRoughness = 0.300$, and $BaseColor = (R = 0.895, G = 0.713, B = 0.602)$. No subsurface scattering was set.

The number of samples per pixel and number of bounces for each of the three lighting passes are listed in Table 6.1.

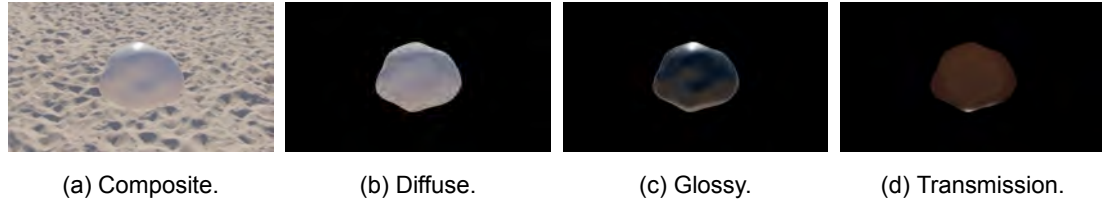


Figure 6.6: Rendering of single grain using the BSDF in Figure 6.5.

Samples per pixel	Bounces		
	diffuse	glossy	transmission
1024	4	4	12

Table 6.1: Number of samples per pixel and number of bounces used for pathtracing in Blender.

The geometry was rendered at three different angles at two different distances from the camera. This is illustrated in Figure 6.7. The geometry was placed in origo, and three cameras were placed at three different elevation angles, two cameras at 25° elevation in opposite directions along the x -direction, *Camera 1* and *Camera 3*, and one camera at 90° elevation angle, *Camera 2*. Their distance from the geometry and origo was varied in accordance to two hemispheres around origo with radii 4 and 16. This was done to observe the appearance from multiple camera angles and distances.

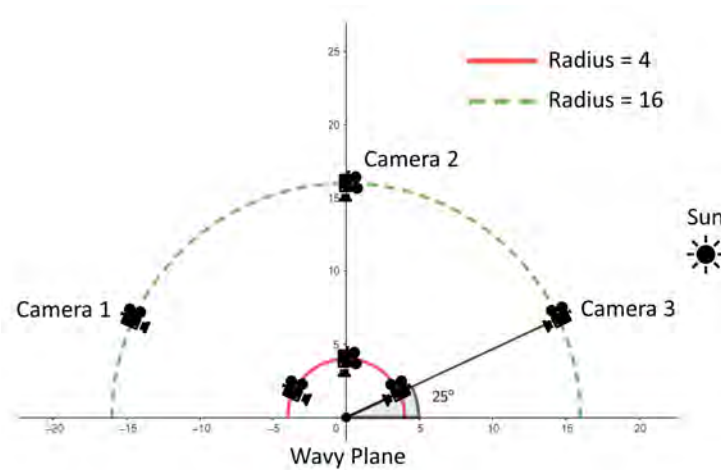


Figure 6.7: Camera setup for render comparisons. As there is no variance in the z -direction between the cameras, sun, and geometry, the setup is illustrated in $2D$. The wavy plane was placed in origo. Three different cameras were used, two at 25° elevation angles in opposite directions along the x -directions in relation to the geometry at origo (*Camera 1* and *Camera 3*), and one camera at 90° elevation angle (*Camera 2*). The cameras were placed at two different distances to the geometry, 4 and 16. The sun annotation merely represents the direction of the sun in the HDRI and does not represent the actual position. The rotation of the HDRI and position of sand geometry was kept the same as the pathtracing experiments, and the sun is at elevation angle 24.70° compared to the sand geometry in origo.

In Unity, the same setup was used, and values for the shader can be seen in Figure 6.8. The same color was used for all sand grains, as this was the case for the pathtracing implementation. P was used to model the porosity of the sand that can be seen in Blender due to the distance between the sand grains. The input to the noise generation function was multiplied to make the size of each grid cell approximate the size of a sand grain in Blender. Besides this, all other variables including base color, subsurface scattering, roughness, transmission, and transmission were identical to the ones chosen for the BSDF in Blender.

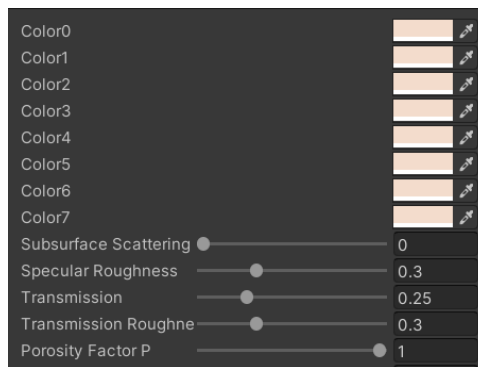


Figure 6.8: Shader parameter values for the shader was set to match the ones of the BSDF in Blender, presented in Figure 6.5.

The results for rendering the explicit sand geometry scene using pathtracing in Blender and the same scene replacing explicit sand geometry with the shader implementation of this project, using forward rendering in Unity, can be seen in Figures 6.9, 6.10 and 6.11. As described, there is a color difference between render results from Blender and Unity, even more pronounced in the results of this section than the previous. While it affects the results considerably, comparison between the HDRI and the rendered sand can still be made within each render, and the overall expression of the contributions can still be compared between Blender and Unity. The render times in ms / frame can be seen in Tables 6.2 and 6.3.

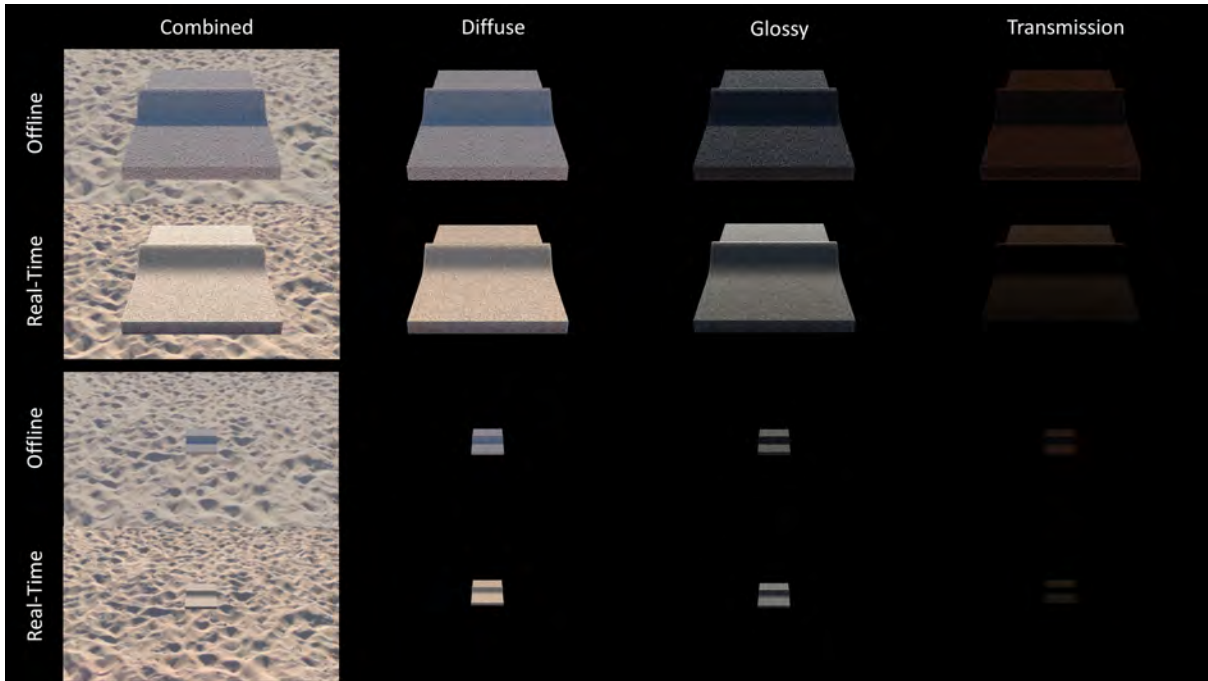


Figure 6.9: Offline results from Blender and real-time results from Unity for Camera 1 for the two different distances. Combined represents the final shading results, diffuse the diffuse contribution, glossy the specular contribution, and transmission the contribution from transmission.

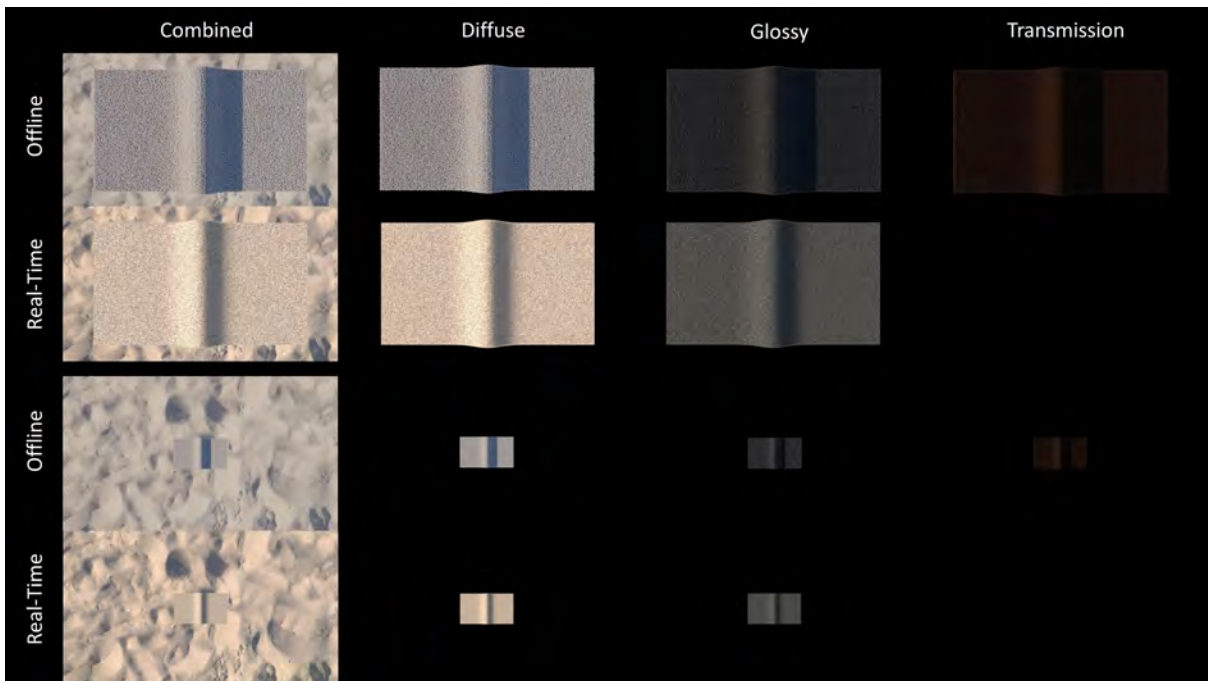


Figure 6.10: Offline results from Blender and real-time results from Unity for Camera 2 for the two different distances. Combined represents the final shading results, diffuse the diffuse contribution, glossy the specular contribution, and transmission the contribution from transmission.

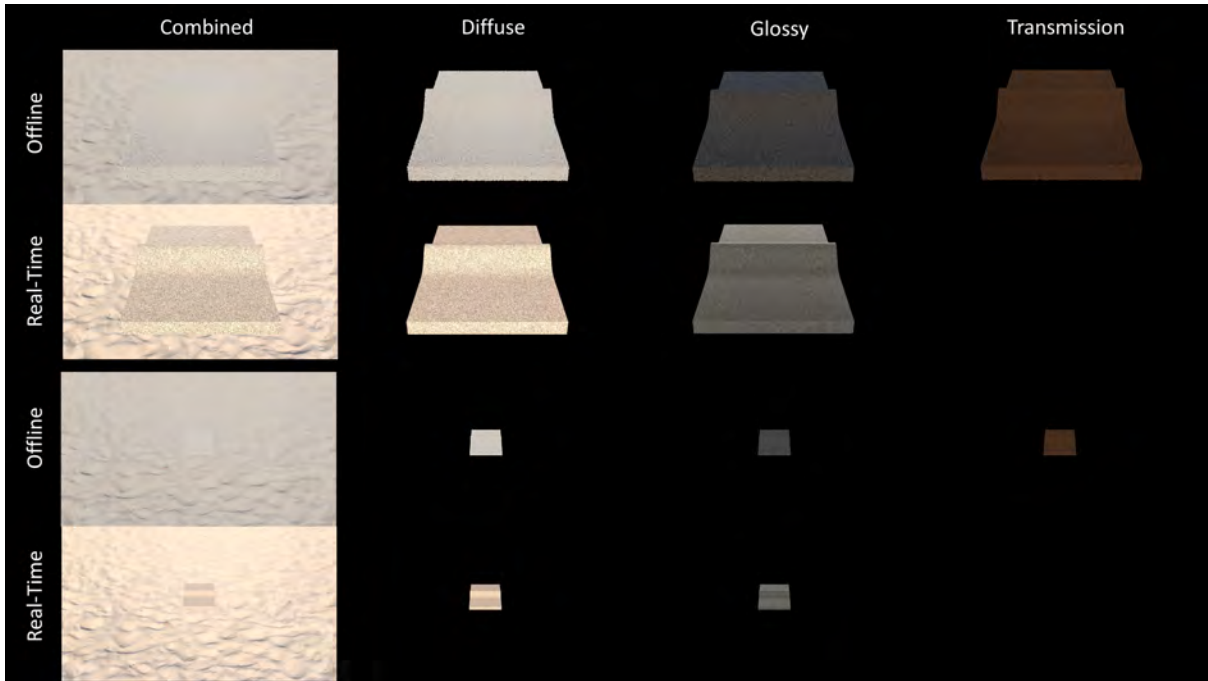


Figure 6.11: Offline results from Blender and real-time results from Unity for Camera 3 for the two different distances. Combined represents the final shading results, diffuse the diffuse contribution, glossy the specular contribution, and transmission the contribution from transmission.

radius	Blender			Unity		
	camera 1	camera 2	camera 3	camera 1	camera 2	camera 3
4	735.09K ms	1602.74K ms	915.75K ms	1.3 ms	1.2 ms	1.3 ms
16	50.43K ms	101.15K ms	55.73K ms	1.3 ms	1.2 ms	1.3 ms

Table 6.2: Comparison of render times in milliseconds (ms) per frame between Blender and Unity when rendering a similar scene. The render times in Unity is an average over frames. In Blender, pathtracing was used to render sand grains distributed through a wavy plane geometry, whereas in Unity forward rendering was used to render the wavy plane geometry with the custom shader of this project. The results not surprisingly show that forward rendering in Unity is faster, but the results also show that the implemented shader does run in real-time in Unity at satisfactory speed.

Average Render Times	
Blender	Unity
576.81 K ms ($\tilde{17.34} \cdot 10^{-7} fps$)	1.27 ms ($\tilde{790} fps$)

Table 6.3: Average render times in milliseconds per frame (ms) and frames per second (fps) for Blender and Unity.

6.2.1 Discussion

As described, the color difference between Blender and Unity and the lack of shadows affects the presented results. The following discussion will compare the offline and real-time renders presented in this section and discuss how well the real-time results approximate the offline pathtracing results. The color difference and lack of shadows will be disregarded for the purpose of further comparing and discussing the results.

Looking at the comparison results, one of the most noticeable differences are the results for transmission. As transmission is only modelled for conditions in which the main light source is on opposing sides of the observer in relation to the geometry, it fails to capture most cases. Even in the cases it does model, it does not convincingly represent the transmissive pass for the offline results. The diffuse and specular passes more convincingly approximate the offline results. While the porosity for the real-time implementation in some cases captures the illumination differences across the surface in the offline results quite well, it can be noticed that it falls short in some. It does not depend on the direction of the main light source, a property that can be observed for the offline results. When the collection of grains is directly illuminated, as in Figure 6.11, the porosity becomes less observable. While the method for handling porosity in this project works well for some cases, compared to the offline results, it will not look believable when observing the material very closely. At the distance depicted in the images, it looks convincing, but zooming in on the images or making renders at closer distances reveals it is simply intensity differences in a grid structure. At greater distances where the shader transitions to shading using a BRDF, the real-time results quite convincingly approximate the appearance of the sand in the HDRI. The process of generating the results in this section revealed how difficult it is to accurately model the appearance of sand from multiple viewing angles even when using accurate offline techniques. The presented offline approach does not appear convincingly like the sand in the HDRI at all angles. The one that is most convincing is seen in Figure 6.11, where the sand completely blends with the background. This suggests that while the offline pathtracing results were great for having a frame of reference, an implementation benefits from considering not only the offline frame of reference religiously, but using this in conjunction with qualified estimates from observing real sand to derive appropriate methods.

6.3 Breakdown of Shading Contributions

The presented shader was used to shade a dunes mesh object by Rostenbach [29] using an HDRI skybox by rpgwhitelock [30]. This allowed for observing the shader across multiple scales in the same rendered image. This section shows how each of the shading contributions affect the final result. Results for each contribution to L_{close} can be seen in Figure 6.12 and the results for each contribution to L_{far} can be seen in Figure 6.13. Figure 6.13 shows the results for combinations of all contributions for L_{close} and L_{far} as well as a debug view of the sigmoid function used to transition between the two based on distance to the camera. The final combined result can be seen in Figure 6.15. The images are provided in high quality and zooming allows for inspecting the contributions further.



(a) Diffuse.

(b) Specular.

(c) Transmission.

Figure 6.12: Shading contributions for L_{close} .



(a) Diffuse.

(b) Specular.

(c) Transmission.

Figure 6.13: Shading contributions for L_{far} .



(a) Close.

(b) Far.

(c) Sig transition.

Figure 6.14: Combination of all shading contributions for L_{close} and L_{far} . The right most image shows the transition function used to transition between L_{close} and L_{far} .

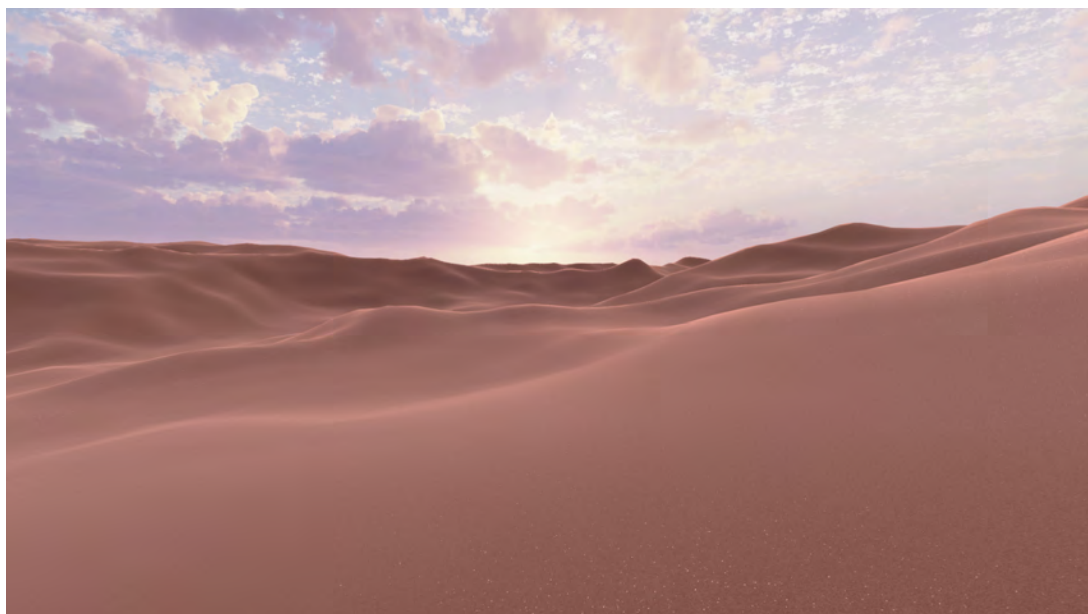


Figure 6.15: Final shading result.

The dunes 3D model consists of 160.5k triangles and 80.8k vertices. Rendering the scene presented in this section where the geometry takes up most of the screen renders on average at 1.9 ms per frame (526.32 fps).

6.3.1 Discussion

The results from this section breaks down the different components in a more game-like environment, where the shaded mesh takes up most of the rendered image. The scene efficiently renders real-time. Compared to the results from other sections, it shows the transitioning from micro to macro scale shading. The exact transition is shown in Figure 6.14c. It shows how the first dune in the image will be shaded completely by the micro shading, and the dunes at increasing distances approach the BRDF shading. It highlights the need for introducing L_{far} , as looking at the image in Figure 6.14a shows that shading using L_{close} only preserves granularity at too great distances which is inaccurate.

6.4 Test Builds

Three WebGL builds were made to allow for exploration of the shader in real-time in different ways.

The first build shows a sphere with the shader attached, and the camera spins around it. There are three radii that can be toggled using the number keys 1,2,3. The build can be accessed [here](#).

The second build was made for the dunes scene presented in the previous section. One cannot experiment with the different user parameter values, however one can get the experience of running around in the sand in a game-like environment. The camera can be translated using the *WASD* keys and rotated using the mouse. The build can be accessed [here](#).

The final build was made for a mesh that approached the shape of the sand in the HDRI [10] used during the pathtracing experiments. This scene was made to show the appearance of the shader on a mesh that further approaches beach sand. The mesh was created by modifying a Blender file provided by Jakob Andreas Bærentzen. As for the dunes build, the camera can be translated using the *WASD* keys and rotated using the mouse. The build can be accessed [here](#).

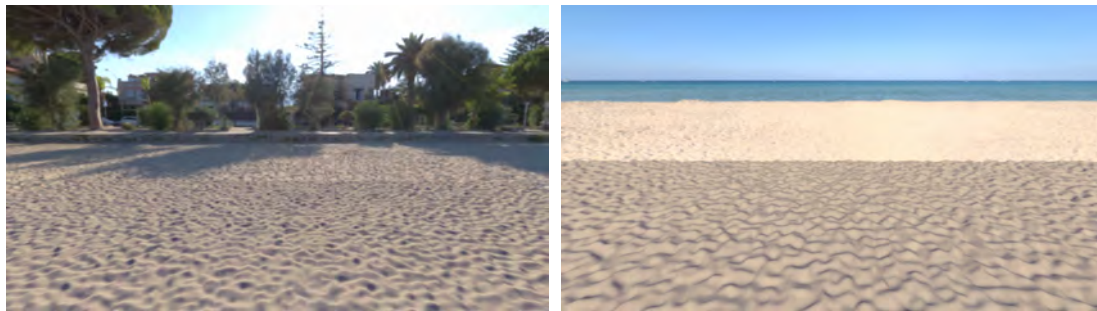
6.4.1 Discussion

The builds allow for experiencing the shader in real-time at multiple distances. A very noticeable aspect that they reveal is the need to consider anti-aliasing techniques. For instance, the glints were jittery, suggesting temporal anti-aliasing is appropriate. Furthermore, especially apparent for the beach scene, noise is introduced in the image at larger distances. This can be observed as bright spots in Figure 6.16. For the beach scene, it shows that the shader does not accurately capture the brightness differences occurring on real sand for varying light source directions. This can be seen in Figure 6.17. Figure 6.17a shows the mesh being back lit and much more closely resembles the sand in the HDR image compared to 6.17b, showing the mesh being

front lit, where the result is much too dark.



Figure 6.16: Noise observed at increasing distances to the camera due to lack of anti aliasing.



(a) Back lit.

(b) Front lit.

Figure 6.17: Limitations of the shader observed in the beach scene. Compared to the real sand in the HDRI, the shader does not accurately represent the intensity differences for when the sand is illuminated from the front and the back in relation to the observer.

6.5 Company Feedback

The results presented in this section was presented for Playdead for feedback in terms of how the current implementation satisfies their implementation needs.

It was expressed that what is especially valuable for the company regarding this project is the detailed analysis and discussion of a specific problem case. A typical workflow for the company includes developing many different types of prototypes before settling on an approach, a process which can be aided by the work that goes into a project such as this.

When presented with the choice of exposed user parameters, they were deemed intuitive for the current approximations. However, they described that the final choice should be made in an iterative process between developer and the artist.

It was suggested investigating how appropriate the use of LOD transitions is in terms of performance for a real-time application, something that has not been investigated in this project. Currently, calculations for both shading at close distances and far distances are being made for each fragment which could result in a high cost in performance. Further experiments for how to use shading LODs and whether it is appropriate for real-time applications could be made.

Additionally, during the project Playdead expressed a desire for the development of a BRDF model that can alleviate the issues they have experienced with Oren-Nayar while keeping the

simplicity of using a BRDF method compared to more expensive models. Analysing and experimenting with creating an improved BRDF model as an alternative to Oren-Nayar could therefore further aid the company in the development of their final approach.

7 Discussion

The results presented in the previous section revealed how well the methods of this project can be used to provide real-time render results that approximate real sand and realistic offline pathtracing renders. The results showed that especially contributions related to light paths that are not accessible for a real-time solution were difficult to model. For instance, for the transmission contribution there is no access to information about the origin of a transmitted ray, meaning accurate approximations of the contributions are difficult to achieve. Contributions that rely on direct light paths that are accessible for the real-time model are much more closely approximated, such as diffuse and specular contributions. The build of the beach scene in which a mesh that models the irregular beach sand from the HDRI is shaded with the real-time shader of this project revealed that for certain light angles, the shader of this project convincingly appears like the real sand in the HDR image. The builds furthermore revealed limitations to the current methods that could only be observed in real-time for moving observer positions, such as the need for utilizing anti-aliasing techniques to further approximate reality.

The approach used in this project in which qualitative assessment is used to approximate offline pathtracing results in real-time is useful but has its limitations. Not only is qualitative assessment prone to errors, the frame of reference only provides a discrete representation of the complex appearance of granular materials, limited by among other geometry, shading, observer position, and chosen lighting. In some cases, the real-time tool-box is substantially more limited than pathtracing techniques, and instead of attempting to model the offline results as accurately as possible, the effort is potentially better spend using the knowledge of what physically takes place and providing an educated approximation that considers the real-time possibilities. This is for instance the case for transmission. While the transmission contribution of this project doesn't completely model the offline transmission contribution, the results of the implementation provides an intuitive transmission contribution that creates plausible results for certain cases, such as for the dunes scene. Here, transmission contributions are present at the top of the dunes when the light hits the dune from behind, something that intuitively makes sense as this models the lower density of grains between the light and the observer where light can travel through more easily. Furthermore, the results of offline pathtracing is not a completely perfect representation of real materials, and therefore comparing to real-life observations in conjunction with offline results is valuable. Choices were also made that were not physically accurate but made as an attempt to model real-life observations. This was the case for glints that were multiplied by an extra intensity value, as they appeared to weak compared to the observations made during the field trip. An offline frame of reference, real-life observations, and knowledge of the physical occurrences behind the shading properties was used in conjunction to create the solution of this project, a combination that arguably makes the final result stronger than relying on just one of the three alone.

The solution of this project was guided to adhere to the requirements from Playdead. The solution does run efficiently in real-time even for meshes that take up most of the screen. Sand was represented at multiple scales, however inspecting the sand closely reveals the grid cells utilized for representing granularity of the material. Most likely, observing the sand at very close scales will not occur often in a game, however it is a limiting factor. In terms of representing the surface to allow for transitioning to particle representation, the solution currently does not include particle representations, however decisions have been made throughout to enable the transition. Each grid cell represents a cube geometry, a rotation, and a color. Particles can detach from each grid cell represented by the given properties and be shaded like the macro surface where macro normals are now represented by the cube normals. The solution attempted to solve the issues the company had experienced with existing micro-facet reflection models by introducing additional macro shading contributions such as Fresnel reflectance and transmission and using low variance values for the micro-facet BRDF. However, the company expressed a desire for development of a BRDF model that can alleviate the experienced issues of Oren-Nayar. Looking into how this can be achieved should be considered for further satisfying the needs of the company. Furthermore, observing the builds revealed the importance of introducing anti-aliasing techniques, something that has not been covered in this project but is something that is arguably very desirable for a games application.

While this project dealt specifically with methods for implementing a real-time representation of sand, the results can be generalized to other granular materials as well. The user parameters can be used to control the appearance of the granular material relating to the color of the granules, subsurface scattering, specularity, transmission and porosity, which most likely to a degree can be used to approximate the appearance of other granular materials. However, thinking of other granules, such as snow, salt, spices etc., they especially differ in their geometric appearance. This project does not deal with how varying geometric representations of individual granules affect the appearance of the granular material, and this can be a limitation for representing other granular materials. For instance, the granules of snow are considerably different than sand. Furthermore, the sand granules were modelled using very simple cube geometry, which does not realistically represent actual sand grains, and it has not been tested whether this is an appropriate approximation for representing sand granules. The project does show how varying material properties of individual granules affect the appearance at macro scale, and this knowledge can be generalized to many other granular materials.

Not all aspects that relate to convincingly rendering granular materials have been investigated in this project. An aspect that has not been considered is the influence of the shape of the macro mesh on the appearance of granular materials. Surfaces of granular materials are rarely smooth, due to porosity and stacking of individual granules arising from friction between granules. This is especially apparent looking at the images from the field trip, where the footprints in the sand and clumps of sand dominate the appearance. It was also apparent from the ripples in the dunes in the appearance study. Modelling finer details of the macro structure could

be appropriate to address in the shader rather than explicitly on the mesh, and a procedural approach could be used that also allow for artist directability in which the artist can guide the representation of details on the macro mesh. The implementation of Journey by thatgame-company [18] for instance considered macro shape details by introducing height map details, modelling ripples in sand, something that could inspire this project. Addressing this could potentially allow for better representations of the granular materials and more types of granular materials.

Generally, this project dealt with answering a problem statement for how a real-time rendering approach be used to approximate the appearance of sand at multiple scales, accounting for its relevant material properties. The phrasing of this problem statement is very open, meaning there is not just one answer as to how this can be done. This project provided one solution to the problem statement while there are countless opportunities for how it could be answered, and it is likely that other types of solutions could offer better answers, something that has not been investigated and compared during this project. Overall, methods for approximating the appearance of sand at multiple scales while considering its relevant shading contributions have been presented in this project, ultimately providing an answer to the problem statement.

8 Conclusion

This project set out to answer the following problem statement.

How can a real-time rendering approach be used to approximate the appearance of sand at multiple scales, accounting for its relevant material properties?

This project dealt with real-time rendering of granular materials, however focused on rendering of sand to allow a more in-depth analysis and method derivation. An appearance study was conducted using photos and videos of sand and realistic offline pathtracing renders of sand were created. The pathtracing results were used to understand the relevant material properties of sand and how these influence its appearance. The appearance study and offline results created a frame of reference for the real-time implementation of this project to assist in answering the problem statement.

The project was done in collaboration with the game company Playdead, and its solution was guided towards being suitable for games applications. The methods of this project represented the granular material at micro and macro scale, micro scale referring to the distribution of sand grain normals across the surface and macro scale as the normals of the surface. The micro structure was modelled in a grid cell arrangement across the surface in object space, where each grid cell was represented by cube normals. The normals were rotated using on pseudo random noise and sampled based the view direction alignment. Shading was divided into two representations; one for close distances that shaded the surface using the micro normals, and one for greater distances where a BRDF was used to model the averaging effect occurring when a single pixel covers multiple sand grains. A function was introduced for transitioning between the two representations based on the distance to the observer. Diffuse, specular, and transmissive contributions were modelled to account for the relevant shading properties observed during the analysis of sand. The implementation considered artist directability by introducing user-controlled parameters for sand grain colors, subsurface scattering, specular roughness, transmission, transmission roughness, and porosity of granules for the material.

This project investigated to which degree a real-time solution can approximate the appearance of sand while dealing with the challenges imposed by real-time models. Compared to other current real-time approaches, this project investigated the connection between offline shader parameters and their influence on the appearance of sand and attempted to recreate the observations in real-time. The results of this project showed that the solution was able to qualitatively approximate expensive offline methods, modelling necessary shading contributions while representing sand at multiple scales in real-time, ultimately providing a way for answering the problem statement of this project.

9 Future Work

One of the very apparent limitations of the current state of the implementation is the lack of anti-aliasing techniques. This is especially apparent for glints. Since glints are calculated using the micro normals that are sampled from randomly rotated cubes, the cube faces in some cases take up less space than a grid cell, meaning the glints contribution occur at a smaller area, rendering single sampling unsuitable quicker. Looking into anti-aliasing techniques, such as temporal anti-aliasing, could potentially alleviate these issues.

Currently, the micro scale is represented by cube normals sampled using pseudo random noise seeded by the fragment position in object space. This however results in a blocky appearance when observing the surface closely. Perhaps another distance would be relevant to consider that deals with the shading at very close distances. Like for the other shading approaches in this project, transitioning to a higher level of detail for very close distances could improve the current solution. However, as suggested by Playdead, it is ideal to look into how performance heavy it is to have different representations and transitioning between them, and therefore introducing an extra scale could be suboptimal. Another approach could be to look into generating noise that does not result in a blocky appearance.

This project has only touched upon a number of elements necessary to represent granular materials. As discussed, this project currently doesn't deal with surface shape details on the macro mesh, however when observing granular materials this seems to be an important aspect to cover. This can be granules clumping together due to friction creating subtle but noticeable height differences across the surface, erosion, wind related appearance expressions, etc. Granular materials can be further affected by their environment in ways that relate to the shading of the material. This is for example the case if the material becomes wet, which result in a distinctively different appearance for the material. This can be imagined thinking of wet beach sand that looks darker and more reflective than dry beach sand, spices that have been subjected to moist and start clumping, etc., another aspect to cover for the solution to further approximate reality.

As experienced by Playdead, current micro facet models such as Oren-Nayar have their limitations. The company expressed a desire for developing a new BRDF model that alleviate these limitations and can provide appropriate shading results for the granular materials they are in need of. Further looking into BRDF representations could be done, as issues were likewise experienced during this project with Oren-Nayar. BRDF methods are also typically more suitable for real-time applications, as they can be more efficient than using complex shader calculations. A development process of a new BRDF method should still regard the findings of this report to ensure the elements necessary for representing sand are still considered.

Finally, this project currently only supports the surface representation of sand. Playdead desired

a surface representation that allowed for individual grains to transition to particle representation. The current methods of this project allow for this transition by representing individual grains across the surface by a position, set of normals, and rotation, however explicitly looking into particle representations and transitions could be part of future versions of this project.

Bibliography

- [1] Gharbalah Industrial Company. (2019). "Silica sand." [Image]. Retrieved from: <https://www.bm.com.sa/product/silica-sand-industrial-raw-materials/> (Accessed 08/01/2022).
- [2] M. Hare. (2019). "Recipe for a dune: Sand, wind, water, plants." [Image]. Retrieved from: <https://science.oregonstate.edu/IMPACT/2019/08/recipe-for-a-dune-sand-wind-water-plants> (Accessed 08/01/2022).
- [3] 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, no. 4, Jul. 2015. DOI: 10/gfzndr.
- [4] E. d'Eon, "An analytic BRDF for materials with spherical lambertian scatterers," *CoRR*, vol. abs/2103.01618, 2021. arXiv: 2103.01618. [Online]. Available: <https://arxiv.org/abs/2103.01618>.
- [5] M. Oren and S. K. Nayar, "Generalization of lambert's reflectance model," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '94, New York, NY, USA: Association for Computing Machinery, 1994, pp. 239–246, ISBN: 0897916670. DOI: 10.1145/192161.192213. [Online]. Available: <https://doi.org/10.1145/192161.192213>.
- [6] S. Sepp, *Sand types*, [Image]. Retrieved from: <https://www.sandatlas.org/sand-types/> (Accessed 22/01/2022), 2011.
- [7] R. Waayers. (2016). "A diversion to the mojave national preserve, part 1: The kelso dunes." [Image]. Retrieved from: <http://cwwildernessjournal.blogspot.com/2016/04/a-diversion-to-mojave-national-preserve.html> (Accessed 03/01/2022).
- [8] A. Hasanova. (2020). "Kelso dunes, california and grand canyon national park, arizona." [Image]. Retrieved from: <https://endlessloopphotography.com/2015/10/12/kelso-dunes-california-and-grand-canyon-national-park-arizona/> (Accessed 03/01/2022).
- [9] Blender Online Community, *Blender - a 3d modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022. [Online]. Available: <http://www.blender.org>.
- [10] A. Mischok. (2020). "Spiaggia di mondello." [HDR]. Retrieved from: https://polyhaven.com/a/spiaggia_di_mondello (Accessed 03/01/2022).
- [11] Blender Foundation. (2019). "Principled bsdf." Available: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/principled.html (Accessed 03/01/2022).
- [12] Shaw Resources. (2019). "What is silica sand how is it different from regular sand?" [Image]. Retrieved from: <https://shawresources.ca/what-is-silica-sand/> (Accessed 03/01/2022).
- [13] Wikipedia. (2021). "List of refractive indices." Available: https://en.wikipedia.org/wiki/List_of_refractive_indices (Accessed 03/01/2022).

- [14] Athabasca Minerals Inc. (n.d.). "Prosvita sand project." [Image]. Retrieved from: <https://athabascaminerals.com/silica-sand/prosvita-1/> (Accessed 19/01/2022).
- [15] Blender Foundation. (2019). "Passes." Available: <https://docs.blender.org/manual/en/latest/render/layers/passes.html> (Accessed 03/01/2022).
- [16] T. Zirr and A. S. Kaplanyan, "Real-time rendering of procedural multiscale materials," in *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ser. I3D '16, Redmond, Washington: Association for Computing Machinery, 2016, pp. 139–148, ISBN: 9781450340434. DOI: 10.1145/2856400.2856409. [Online]. Available: <https://doi.org/10.1145/2856400.2856409>.
- [17] thatgamecompany, *Journey*, 2020. [Online]. Available: <https://thatgamecompany.com/journey/>.
- [18] J. Edwards, *Sand Rendering in Journey*, <https://www.gdcvault.com/play/1017742/Sand-Rendering-in>, 2013.
- [19] M. Jarzynski and M. Olano, "Hash functions for gpu rendering," *Journal of Computer Graphics Techniques (JCGT)*, vol. 9, no. 3, pp. 20–38, Oct. 2020, ISSN: 2331-7418. [Online]. Available: <http://jcg.org/published/0009/03/02/>.
- [20] P. Dutré, *Global Illumination Compendium*. 2003, Produced at the Program of Computer Graphics, Cornell University.
- [21] J. R. Frisvad, "Building an orthonormal basis from a 3d unit vector without normalization," *Journal of Graphics Tools*, vol. 16, no. 3, pp. 151–159, 2012. DOI: 10.1080/2165347X.2012.689606. eprint: <https://doi.org/10.1080/2165347X.2012.689606>. [Online]. Available: <https://doi.org/10.1080/2165347X.2012.689606>.
- [22] J. A. Bærentzen, M. Nobel-Jørgensen, J. R. Frisvad, and N. J. Christensen, *Hands on real-time graphics*, Lecture Notes, Technical University of Denmark., n.d.
- [23] C. Schlick, "An inexpensive brdf model for physically-based rendering," *Computer Graphics Forum*, vol. 13, no. 3, pp. 233–246, 1994. DOI: <https://doi.org/10.1111/1467-8659.1330233>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.1330233>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.1330233>.
- [24] Engineering Statistics Handbook. (2018). "Normal distribution." Available: <https://www.itl.nist.gov/div898/handbook/eda/section3/eda3661.htm> (Accessed 12/01/2022).
- [25] M. Saeed. (2021). "A gentle introduction to sigmoid function." Available: <https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/> (Accessed 06/01/2022).
- [26] B. T. Phong, "Illumination for computer generated pictures," *Commun. ACM*, vol. 18, no. 6, pp. 311–317, Jun. 1975, ISSN: 0001-0782. DOI: 10.1145/360825.360839. [Online]. Available: <https://doi.org/10.1145/360825.360839>.
- [27] M. Pharr, J. Wenzel, and G. Humphreys, "Microfacet models," in *Physically Based Rendering: From Theory To Implementation*, 3rd. 2018.
- [28] Unity Technologies, *Unity - the platform of choice for multiplayer hits*, Unity IPR, Copenhagen, 2022. [Online]. Available: <https://unity.com/>.

- [29] Rostenbach. (2021). "Dunes (wip, workflow test)." [3D Model]. Retrieved from: <https://sketchfab.com/3d-models/dunes-wip-workflow-test-58bd8be249944f1b892a52eb52e06024> (Accessed 13/01/2022).
- [30] rpgwhitelock. (2021). "Allsky free - 10 sky / skybox set (version 1.1.0)." [Cubemap]. Retrieved from: <https://assetstore.unity.com/packages/2d/textures-materials/sky/allsky-free-10-sky-skybox-set-146014> (Accessed 17/01/2022).

A Appendix 1 Shader Implementation

```
1 Shader "Custom/Sand" {
2   Properties{
3     // User Defined Variables
4     _Color0("Color0", Color) = (1,1,1,1)
5     _Color1("Color1", Color) = (1,1,1,1)
6     _Color2("Color2", Color) = (1,1,1,1)
7     _Color3("Color3", Color) = (1,1,1,1)
8     _Color4("Color4", Color) = (1,1,1,1)
9     _Color5("Color5", Color) = (1,1,1,1)
10    _Color6("Color6", Color) = (1,1,1,1)
11    _Color7("Color7", Color) = (1,1,1,1)
12    _SSS("Subsurface Scattering", Range(0,1)) = 0.8
13    _Rs("Specular Roughness", Range(0,1)) = 0.12
14    _T("Transmission", Range(0,1)) = 0.8
15    _Rt("Transmission Roughness", Range(0,1)) = 0.2
16
17    // Debug Variables
18    _Sigma("Glints Sigma", Range(0.000001, 1)) = 0.00001
19    _P("Porosity Factor P", Range(0,1)) = 0.1
20    _NoiseScale("Noise Scale", Range(1,1000000)) = 2000
21    [HDR]_Iglints("Glints Color", Color) = (1,1,1,1)
22  }
23  SubShader{
24
25    Tags {
26      "RenderType" = "Opaque" "Queue" = "Geometry" "RenderPipeline" = "
27      UniversalPipeline" "LightMode" = "UniversalForward"
28    }
29
30    Pass {
31      Name "FORWARD"
32      Cull Off
33      CGPROGRAM
34      #pragma vertex vert
35      #pragma fragment frag
36      #define UNITY_PASS_FORWARDBASE
37      #include "UnityCG.cginc"
38      #include "AutoLight.cginc"
39      #include "Lighting.cginc"
40      #pragma target 3.0
41
42      // User defined variables
43      float4 _Color0; float4 _Color1; float4 _Color2; float4 _Color3;
44      float4 _Color4; float4 _Color5; float4 _Color6; float4 _Color7;
45      float _SSS;
46      float _Rs;
47      float _Rt;
48      float _T;
49      float _P;
50
51      // Debug variables
52      float4 _Iglints;
```

```

52     float _Sigma;
53     float _NoiseScale;
54
55     struct VertexInput {
56         float4 vertex : POSITION;
57         float3 normal : NORMAL;
58         float2 texcoord0 : TEXCOORD0;
59         float2 texcoord1 : TEXCOORD1;
60     };
61
62     struct VertexOutput {
63         float4 pos : SV_POSITION;
64         float2 uv0 : TEXCOORD0;
65         float2 uv1 : TEXCOORD1;
66         float3 normalDirection : TEXCOORD3;
67         float3 worldPos : TEXCOORD4;
68         float3 objectPos : TEXCOORD5;
69         LIGHTING_COORDS(6,7)
70     };
71
72     //-----
73     //helper functions
74
75     float length(float3 v) {
76         return sqrt(sqr(v.x) + sqr(v.y) + sqr(v.z));
77     }
78
79     int colorIndex(float3 rand) {
80         float x = (rand.x + 1) / 2;
81         float y = (rand.y + 1) / 2;
82         float z = (rand.z + 1) / 2;
83         x = x >= 0.5 ? 1 : 0;
84         y = y >= 0.5 ? 1 : 0;
85         z = z >= 0.5 ? 1 : 0;
86         return 4 * x + 2 * y + z;
87     }
88
89     float3 averageColor() {
90         return (_Color0 + _Color1 + _Color2 + _Color3 + _Color4 + _Color5 + _Color6 +
91             _Color7) / 8;
92     }
93
94     //-----
95     //random number generation
96
97     // Jarzynski et al. 2020
98     int3 pcg3d(float3 s)
99     {
100         int3 v = int3(s.x, s.y, s.z);
101         v = v * 1664525 + int3(1013904223, 1013904223, 1013904223);
102         v.x += v.y * v.z; v.y += v.z * v.x; v.z += v.x * v.y;
103         v.x ^= v.x >> 16; v.y ^= v.y >> 16; v.z ^= v.z >> 16;
104         v.x += v.y * v.z; v.y += v.z * v.x; v.z += v.x * v.y;
105         return v;
106     }
107

```

```

108 //-----
109 //sampling of normals
110
111 // Frisvad 2012
112 float3 rotate_to_normal(float3 normal, float3 v)
113 {
114     float sign = 1;
115     if (normal.z < 0) sign = -1;
116     float a = -1.0f / (1.0f + abs(normal.z));
117     float b = normal.x * normal.y * a;
118     v = float3(1.0f + normal.x * normal.x * a, b, -sign * normal.x) * v.x
119         + float3(sign * b, sign * (1.0f + normal.y * normal.y * a), -normal.y) * v.y
120         + normal * v.z;
121     return v;
122 }
123
124 float3 getCubeNormal(int index)
125 {
126     float3 cube[] = {
127         float3(1, 0, 0),
128         float3(0, 1, 0),
129         float3(0, 0, 1),
130         float3(-1, 0, 0),
131         float3(0, -1, 0),
132         float3(0, 0, -1)
133     };
134     return cube[index];
135 }
136
137 float3 getViewAlignedNormal(float3 viewDir, float3 q) {
138     int index = 0;
139     float maxDot = -100;
140     float dotP = 0;
141
142     for (int y = 0; y < 6; y++)
143     {
144         dotP = dot(viewDir, getCubeNormal(y));
145         if (dotP > maxDot) {
146             maxDot = dotP;
147             index = y;
148         }
149     }
150     return getCubeNormal(index);
151 }
152
153 float3 sampleNormalInSphere(float2 rand)
154 {
155     rand.x = (rand.x + 1.0) / 2.0;
156     rand.y = (rand.y + 1.0) / 2.0;
157
158     float phi = 2.0 * 3.141592653589 * rand.x;
159     float x = 2 * cos(phi) * sqrt(rand.y * (1 - rand.y));
160     float y = 2 * sin(phi) * sqrt(rand.y * (1 - rand.y));
161     float z = 1 - 2 * rand.y;
162     float3 q = float3(x, y, z);
163
164     return q;

```

```

165     }
166
167     // Frisvad 2012
168     float3x3 inverseMTM(float3 n) {
169         float3 b1, b2;
170         if (n.z < -0.9999999f)
171         {
172             b1 = float3(0.0f, -1.0f, 0.0f);
173             b2 = float3(-1.0f, 0.0f, 0.0f);
174         }
175         else
176         {
177             float a = 1.0f / (1.0f + n.z);
178             float b = -n.x * n.y * a;
179             b1 = float3(1.0f - n.x * n.x * a, b, -n.x);
180             b2 = float3(b, 1.0f - n.y * n.y * a, -n.y);
181         }
182
183         float3x3 mtm = float3x3(b1,b2,n);
184         return mtm;
185     }
186
187     float3 getMicroNormal(float3 rnd, float3 viewDirection) {
188         float3 q = sampleNormalInSphere(rnd);
189         float3x3 imtm = inverseMTM(normalize(q));
190         float3 objectViewDir = normalize(mul(imtm, viewDirection));
191         float3 objectNormal = getViewAlignedNormal(objectViewDir, q);
192         float3 worldNormal = rotate_to_normal(q, objectNormal);
193         return worldNormal;
194     }
195
196     //-----
197     //shading
198
199     // Schlick 1994
200     float Schlick(float3 v, float3 n) {
201         float ior = 1.458;
202         float f0 = pow(ior - 1, 2) / pow(ior + 1, 2);
203         float F = f0 + (1 - f0) * pow(1 - abs((dot(n, v))), 5);
204         return F;
205     }
206
207     float Cos2Theta(float3 w) {
208         return w.z * w.z;
209     }
210
211     float AbsCosTheta(float3 w) {
212         return abs(w.z);
213     }
214
215     float Sin2Theta(float3 w) {
216         return max(0.0, 1.0 - Cos2Theta(w));
217     }
218
219     float SinTheta(float3 w) {
220         return sqrt(Sin2Theta(w));
221     }

```



```

222
223 float CosPhi(float3 w) {
224     float sinTheta = SinTheta(w);
225     float result = 0;
226     if (sinTheta == 0) {
227         result = 1;
228     }
229     else {
230         result = clamp(w.x / sinTheta, -1, 1);
231     }
232     return result;
233 }
234
235 float SinPhi(float3 w) {
236     float sinTheta = SinTheta(w);
237     float result = 0;
238     if (sinTheta == 0) {
239         result = 0;
240     }
241     else {
242         result = clamp(w.y / sinTheta, -1, 1);
243     }
244     return result;
245 }
246
247 // Pharr et al. 2018
248 float Loren(float3 l, float3 v, float3 n, float sigma) {
249     float sigma2 = sigma * sigma;
250     float A = 1.f - (sigma2 / (2.f * (sigma2 + 0.33f)));
251     float B = 0.45f * sigma2 / (sigma2 + 0.09f);
252     float sinThetaI = SinTheta(l);
253     float sinTheta0 = SinTheta(v);
254     float sinAlpha = 0;
255     float tanBeta = 0;
256     float maxCos = 0;
257     // << Compute cosine term of -OrenNayar model>>
258     if (sinThetaI > 0.0001 && sinTheta0 > 0.0001) {
259         float sinPhiI = SinPhi(l);
260         float cosPhiI = CosPhi(l);
261         float sinPhi0 = SinPhi(v);
262         float cosPhi0 = CosPhi(v);
263         float dCos = cosPhiI * cosPhi0 + sinPhiI * sinPhi0;
264         maxCos = max(0.0, dCos);
265     }
266     // <<Compute sine and tangent terms of -OrenNayar model >>
267     if (AbsCosTheta(l) > AbsCosTheta(v)) {
268         sinAlpha = sinTheta0;
269         tanBeta = saturate(sinThetaI / AbsCosTheta(l));
270     }
271     else {
272         sinAlpha = sinThetaI;
273         tanBeta = saturate(sinTheta0 / AbsCosTheta(v));
274     }
275
276     return (A + B * maxCos * sinAlpha * tanBeta);
277 }
278

```

```

279 // Fresnel strength function
280 float F() {
281     return saturate(-20.80 * pow(_Rs, 5) + 60 * pow(_Rs, 4) - 55.9 * pow(_Rs, 3) +
282         14.55 * pow(_Rs, 2) + 2 * _Rs + 0.25);
283 }
284 // Glints strength function
285 float G() {
286     return saturate(1 / (1 + exp(13.5 * (_Rs - 0.4))));
287 }
288
289 // Diffuse extinction due to transmission
290 float Text() {
291     return saturate(-3.2 * pow(_T, 4) + 4.8 * pow(_T, 3) - 2.2 * pow(_T, 2) - 0.3 * _T
292         + 1);
293 }
294 // Glints Gaussian Distribution Function
295 float GDF(float x) {
296     float mu = 0;
297     return max(0, exp(-pow(x - mu, 2) / (2 * pow(_Sigma, 2))));
298 }
299
300 // Custom transmission function
301 float3 transmission(float3 n, float3 v, float3 l) {
302     float Rt = 0.2 + _Rt * (0.4 - 0.2);
303     float t = 1 - max(0, dot(n, v));
304     t = pow(t, 1 / Rt);
305     t *= max(0, dot(l, -v));
306     t *= _T;
307     t *= (1 - 0.9 * _Rt);
308     return saturate(t);
309 }
310
311 float sig(float x, float a, float b) {
312     return 1 / (1 + exp(-a * (x - b)));
313 }
314
315 // Vertex shader
316 VertexOutput vert(VertexInput v) {
317     VertexOutput o = (VertexOutput)0;
318     o.uv0 = v.texcoord0;
319     o.uv1 = v.texcoord1;
320     o.pos = UnityObjectToClipPos(v.vertex);
321     o.worldPos = mul(unity_ObjectToWorld, v.vertex);
322     o.normalDirection = UnityObjectToWorldNormal(v.normal);
323     o.objectPos = v.vertex;
324
325     TRANSFER_VERTEX_TO_FRAGMENT(o);
326     return o;
327 }
328
329 // Fragment shader
330 float4 frag(VertexOutput i) : COLOR
331 {
332     // General variables
333     float PI = 3.141592653589;

```

```

334
335 // Subsurface scattering variables
336 float scattering = 0.2;
337 float absorption = 0.8;
338 float fs = scattering * _SSS;
339 float fe = exp(-absorption * _SSS);
340
341 // Oren-Nayar variables
342 float orenNayarSigma = 0.1;
343
344 // Sigmoid function variables
345 float a = 2;
346 float b = 1.3;
347
348 // Distance to eye from point
349 float pointDistance = length(i.worldPos.xyz - _WorldSpaceCameraPos.xyz);
350
351 // Pseudo random noise generation
352 float3 rand = normalize(pcg3d(float3(i.objectPos * _NoiseScale)));
353
354 // Normal and light direction calculations
355 float3 viewDirection = normalize(_WorldSpaceCameraPos.xyz - i.worldPos.xyz);
356 float3 macroNormalDirection = i.normalDirection;
357 float3 lightDirection = normalize(lerp(_WorldSpaceLightPos0.xyz,
    _WorldSpaceLightPos0.xyz - i.worldPos.xyz, _WorldSpaceLightPos0.w));
358 float3 microNormalDirection = getMicroNormal(rand, viewDirection);
359 float3 microLightReflectDirection = normalize(reflect(-lightDirection,
    microNormalDirection));
360 float3 macroLightReflectDirection = normalize(reflect(-lightDirection,
    macroNormalDirection));
361 float3 macroViewReflectDirection = normalize(reflect(-viewDirection,
    macroNormalDirection));
362 float cosine_theta = max(0, dot(macroNormalDirection, lightDirection));
363
364 // Unity function for retrieving environment light
365 UnityGI gi_macro = GetUnityGI();
366
367 // Light colors
368 float3 lightColor = _LightColor0.rgb;
369 float3 skyLight = gi_macro.indirect.diffuse.rgb;
370
371 // Sand colors
372 float4 C[] = {
373     _Color0, _Color1, _Color2, _Color3,
374     _Color4, _Color5, _Color6, _Color7
375 };
376
377 float3 rho_m = C[colorIndex(rand)].rgb;
378 float3 rho_n = averageColor();
379 float Imin = saturate(exp(-((_P / pointDistance))));
380 float rho_m_intensity = Imin + (1 - (rand.x * (1 - Imin) + Imin));
381 float rho_n_intensity = Imin + (1 - Imin) / 2;
382 float3 Kd_m = rho_m / PI;
383 float3 Kd_n = rho_n / PI;
384 float3 fresnelColor = skyLight + (lightColor * cosine_theta);
385 float3 glintsColor = (skyLight + lightColor) * _Iglints;
386 float3 transmissionColor = lightColor * rho_n;

```

```

387
388 // DIFFUSE and SUBSURFACE SCATTERING
389 // Diffuse close
390 float3 Ld = Kd_m * lightColor * (fs + (1 - fs) * cosine_theta);
391 Ld += Kd_m * skyLight;
392 Ld *= fe;
393
394 // Diffuse far
395 float3 Lbrdf = lightColor * Kd_n * (fs + (1 - fs)
396 * (Loren(lightDirection, viewDirection, macroNormalDirection, orenNayarSigma)
397 * cosine_theta));
398 Lbrdf += skyLight * Kd_n;
399 Lbrdf *= fe;
400
401 // GLINTS
402 float g = G() * max(0, GDF(rand.x)
403 * dot(viewDirection, microLightReflectDirection)) * cosine_theta;
404
405 // FRESNEL
406 float3 h = normalize(lightDirection + macroLightReflectDirection);
407 float f = F() * Schlick(viewDirection, h);
408 Ld *= (1 - f);
409 Lbrdf *= (1 - f);
410
411 // SPECULAR
412 float3 Lg = glintsColor * g;
413 float3 Lf = fresnelColor * f;
414 float3 Ls = Lg + Lf;
415
416 // TRANSMISSION
417 float t = transmission(macroNormalDirection, viewDirection, lightDirection);
418 float3 Lt = transmissionColor * t;
419 Ld *= Text();
420 Lbrdf *= Text();
421
422 // CLOSE SHADING
423 float3 Lclose = Ld + Ls + Lt;
424
425 // FAR SHADING
426 float3 Lfar = Lbrdf + Lf + Lt;
427
428 // POROSITY
429 Lclose *= rho_m_intensity;
430 Lfar *= rho_n_intensity;
431
432 // TRANSITION CLOSE TO FAR
433 float3 Lfinal = (1 - sig(pointDistance, a, b)) * Lclose
434 + sig(pointDistance, a, b) * Lfar;
435
436 return float4(Lfinal, 1);
437 } ENDCG
438 }
439 }
440 }

```

