Comparison of material models in modern physically based rendering pipelines

Bachelorarbeit
im Studiengang Audiovisuelle Medien
an der Hochschule der Medien Stuttgart

vorgelegt von
Franca Bittner
Matrikelnummer: 33355

am 28. August 2020

zur Erlangung des akademischen Grades eines
Bachelor of Engineering (B.Eng.)

Erstprüfer: Prof. Dr.-Ing. Martin Fuchs
Zweitprüfer: Prof. Dr. Jan Fröhlich

Ehrenwörtliche Erklärung

Hiermit versichere ich, Franca Bittner, ehrenwörtlich, dass ich die vorliegende Bachelorarbeit mit dem Titel: „Comparison of material models in modern physically based rendering pipelines“ selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§26 Abs. 2 Bachelor-SPO (6 Semester), § 24 Abs. 2 Bachelor-SPO (7 Semester), § 23 Abs. 2 Master-SPO (3 Semester) bzw. § 19 Abs. 2 Master-SPO (4 Semester und berufsbegleitend) der HdM) einer unrichtigen oder unvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.

_____ _____

Ort, Datum Unterschrift

Acknowledgements

First, I would like to thank my supervisors, Prof. Dr.-Ing. Martin Fuchs and Prof. Dr. rer. nat. Jan Fröhlich, for inspiring the path of my research, guiding me through each stage of the process and generally enthusing me with science. Thank you for taking all the time in hours-long video conferences to answer each of my questions with your expertise and feedback. This was extremely helpful for me.

Furthermore, I would like to thank the academic staff at the Stuttgart Media University. In particular, I would like to acknowledge Jochen Bomm for his support and insights in developing the rendering scripts for the vast software Maya once again. Moreover, I would like to thank Robin Schulte for providing me with the needed software tools and remote computers just within a few days. As always, I genuinely appreciate your support.

I would also like to thank every other member of the Stuttgart Media University, whether professors, lecturers, technical staff, fellow students, or other. Without you, all the projects I have been involved would not have been conceivable. I am going to miss the time at the HdM Stuttgart very much.

I would like to thank Dr. rer. nat. Frank J. Maile for broadening my perspective on this topic and for giving me an insight into the research network on the subject. Also, I would like to thank Tiantada Hiranyachattada for her kind answers to my e-mails.

Finally, I would like to thank my wonderful parents and lovely sister for always having a sympathetic ear and bearing with me. And I would like to thank my friends for motivating and cheering me up when I needed it. You are the best.

Abstract

The appearance of materials results from a complex interaction of light, material properties and the geometric shape of an object. In the field of computer graphics, many different models have been developed to describe these interrelations. Especially the philosophy of physically based rendering (PBR) is commonly adapted in modern rendering pipelines. This study examines if the reproduction of materials differs across modern physically based rendering tools. More precisely, it compares the intuitiveness of material design as well as the quality and range of reproducible materials among current rendering software.

A sequential rendering framework was developed to evaluate the visual influences of four selected parameters on material appearance. The rendered images are qualitatively compared with the use of material charts, scanline plots and difference images.

The results show that the examined rendering tools mostly yield similar results, with the main differences arising from disparate rendering methods. However, subtle variations between the probed tools are noticeable revealing that every renderer has individual advantages and disadvantages in terms of intuitiveness and physically accuracy.

Kurzfassung

Das Erscheinungsbild von Materialien ist ein komplexes Zusammenspiel aus Licht, Materialeigenschaften und geometrischen Körper eines Objekts. Im Bereich der Computergrafik wurden viele verschiedene Modelle entwickelt, um diese Zusammenhänge zu beschreiben. In aktuellen Rendering-Pipelines wird vor allem die Philosophie des physikalisch basierten Renderings (PBR) häufig adaptiert. Diese Arbeit widmet sich der Fragestellung, ob sich die Wiedergabe von Materialien in modernen physikalisch basierten Renderern unterscheidet. Insbesondere wird die Intuitivität von Materialdesign sowie die Qualität und die Bandbreite an reproduzierbaren Materialien in aktueller Rendering-Software verglichen.

Um die visuellen Einflüsse von vier ausgewählten Parametern auf das Erscheinungsbild des Materials zu evaluieren, wurde ein sequenzielles Rendering-Framework entwickelt. Die gerenderten Bilder werden mit Hilfe von Materialtabellen, Bildzeilen-Diagrammen und Differenzbildern qualitativ verglichen.

Die Ergebnisse zeigen, dass sich die untersuchten Tools weitestgehend ähnliche Ergebnisse liefern und die größten visuellen Unterschiede auf die grundlegende Rendering-Methode zurückzuführen sind. Allerdings sind subtile Unterschiede zwischen allen Renderern zu erkennen, die verdeutlichen, dass jeder Renderer individuelle Vor- und Nachteile in Bezug auf Intuitivität und physikalische Genauigkeit hat.

List of Figures

List of Tables

List of Abbreviations

BRDF	**B**idirectional **R**eflectance-**D**istribution **F**unction
BSDF	**B**idirectional **S**cattering **D**istribution **F**unction
BSSDF	**B**idirectional **S**ubsurface **S**cattering **D**istribution **F**unction
BSSRDF	**B**idirectional **S**cattering-**S**urface **R**eflectance-**D**istribution **F**unction
BTDF	**B**idirectional **T**ransmission **D**istribution **F**unction
BTF	**B**idirectional **T**exture **F**unction
FBX	**F**ilm**b**o**x**
HDRi	**H**igh **d**ynamic **r**ange **i**mage
IOR	**I**ndex **o**f **r**efraction
PBR	**P**hysically **b**ased **r**endering
UE4	**U**nreal **E**ngine **4**
UI	**U**ser **I**nterface
VR	**V**irtual **R**eality

Contents

1 Introduction

In many today's digital production workflows, physically based rendering (PBR) plays a central role in the working of a 3D artist. The main idea of PBR is to reproduce the virtual world based on the laws of physics. This applies to several topics such as the simulation of water or cloth movement, but especially to the shading of materials. While physically based rendering is best known for creating photorealistic looks, various non-photorealistic looks can be achieved as well. To ensure creative workflows, it is crucial to provide an intuitive editing interface for users not being familiar with the underlying physical terminology. The Walt Disney Animation Studios played a leading role in establishing the creative principles of PBR in modern digital production workflows. Burley (2012) described a new material model, which compromises physical laws and the needs of artists for material design. Meanwhile, many other commonly used 3D rendering software, such as Maya, Blender, Unity or Unreal Engine 4 (UE4), have adopted the PBR workflow, and with the application suite Substance by Adobe, there even is a tool that is purely dedicated to design material appearance.

The study of material models is an active field of research. The topic addresses several interdisciplinary areas, such as "psychology, computer graphics, neuroscience [and] industrial design" ('DyViTo Project', 2017). By now, a variety of models for describing material appearance exists in terms of physically based rendering. Yet, these models cannot easily be compared. Although there is already plenty of research on material models and their taxonomy, the quantity of material models can easily get confusing for the user, especially since every rendering tool uses its own parameter set. This becomes a problem as it is common to interchange 3D projects between different rendering tools in modern digital production pipelines, where each department may use its own preferred software (cf. Guarnera et al., 2018). It cannot be assured that this exchange happens without loss of material description data. Certain parameters may not exist in another software or have a different impact on the rendered result, as they are included differently in the rendering equation. Since these dissimilarities are insufficiently studied, this thesis aims to compare the different material models in modern rendering software commonly used in digital productions. It will be investigated how the different material models in current physically based rendering pipelines differ in terms of the intuitiveness of material design and the quality and range of reproducible materials.

1.1 Related Work

Similar to this paper, Guarnera et al. (2018) have dealt with the visual differences occurring when transferring parameter values from one parametric material model to another. The authors developed a genetic algorithm that finds a source parameter set matching the desired original parameters of a different material model by comparing two images, the so-called "BRDF Difference Probe[s]" (Guarnera et al., 2018, p. 1). In an iterative process two parental data sets are combined to generate new possibly fitting parameter sets. In the end, the parameter sets of both material models are matched and evoke a nearly identical output image. While this work is far more sophisticated than this paper will be in the given amount of time, it does not examine material models commonly used in current digital production workflows in detail. This work is intended to fill this gap. The chosen rendering tools are discussed later.

Secondly, Burley (2012), Karis (2013), Lagarde (2011b, 2012b, 2014), Unity Technologies (2014) and the Blender Foundation (2017) provide material charts similar to the results in this work. However, those illustrations are only intended for the respective own rendering tool and do not provide any comparison values among each other. Moreover, there are not material charts for every commonly used rendering tool. For example, the Arnold renderer only provides a few samples of individual material parameters, but no coherent overview over all material parameters (cf. Solid Angle S.L., n.d.-b). As reference images are actually in demand for testing different implementations and ensuring "look consistency" (Moeller & Georgiev, 2020), this work aims to provide a rendering framework to generate such reference material charts across all examined renderers.

Additionally, Burley (2012, 2015), Karis (2013), Burley et al. (2018) and Georgiev et al. (2019a) provide a detailed discussion of their material models used for the PBR rendering pipelines in several digital productions. Their work has been extremely helpful in evaluating the material models of current rendering pipelines, as it has provided reference points for the elaboration of the models used in current renderers.

Another topic related to this work is the improvement of material appearance design. There are three studies to focus attention on. The state of the art report of Schmidt et al. (2014) deals with alternative ways of editing a virtual scene. The authors define the term "appearance design" and describe several concepts that build a bridge between lighting and material editing, as for instance editing the lighting scenario by clicking and dragging a specular highlight of an object (T.-W. Schmidt et al., 2014, pp. 2–5). In accordance with Schmidt et al. (2014), this thesis considers the appearance of a scene as an interplay of lighting and surface materials, or global and local light transport respectively (p. 2, 4, 6, 9). However, in the end the writers note that it "remains for future work, to convey additional information […] about light transport and

material interaction" (p. 9). Although this paper might be outdated already, it shows the importance of understanding the synergy of light transport and material properties, which are discussed in later sections of this paper.
Serrano et al. (2016) developed an intuitive control space for editing captured BRDF data. Therefor the authors asked subjects for perceptual attributes that best describe the appearance of a material. Especially the proposed list of fourteen attributes for building an intuitive parameter set for material design (p. 4) is of great interest for this paper. However, data-driven BRDFs are less relevant at least in the context of digital media productions, since in the commonly used tools the representation and editing of those data is not supported yet and not feasible either due to high memory requirements and expensive calculations.
Finally, Gulbrandsen (2014) depicts an example for mapping unintuitive parameters of a physically plausible model to "artist friendly" (pp. 64-65) parameters by decoupling the influences of two parameters on the appearance of the Fresnel curve. This paper serves as a favourable example for suiting the PBR workflow to the user when assessing the intuitiveness of material design in the chosen rendering tools.

1.2 Terminology

There are two different terms related to the description of material features. The expression *material appearance* is a perceptual quantity and thus strongly linked to the human visual system, whereas the term *material property* describes the physical and chemical characteristics of a material. In the field of computer graphics, *material models* refer to the underlying shading model for calculating the appearance of a material. In modern rendering tools, material models are generally parametrized and thus material appearance can be edited by setting *material parameters*, which are descriptions of either material properties, such as the *Index of Refraction*, or material appearance features like *Glossy*, or both, for example *Metallic* or *Roughness*. In this paper, material parameters will always be written in italics to distinguish the parameter names from naturally used words, such as roughness in contrast to *Roughness*.

The term *material chart* has come to be used for figures that depict the influence of a material parameter over its entire value range with a specific step size. They are used to demonstrate the effect of material parameters, to show off the range of reproducible materials or possibilities of material design and to make statements about the influence of each parameter, for example whether they are linearly parametrized or not. Moreover, material charts are used for validating shader replicas in different rendering pipelines to identify visual disparities and to ensure a consistent look across all renderers (Moeller & Georgiev, 2020).

1.3 Choice of rendering tools

In this paper four renderers commonly used in the media industry are discussed. Both, real-time and offline rendering pipelines are considered, representing different use cases like film production or game development. In particular the chosen rendering tools are Arnold for Maya (Autodesk, 2020), Cycles and Eevee from Blender (Blender Foundation, 2020a) and Unreal Engine 4 (Epic Games, 2020b).

The set of chosen renderers comprises two path tracers that calculate light transport by following the path of a ray cast into the scene. Arnold and Cycles are based on this method and are designated as offline renderers, since this method requires more rendering time but generates "an unbiased target result because it's not limited by the number of samples it can use" (Epic Games, 2020a). Although Unreal is capable of path tracing as well, only the real-time rasterization rendering pipeline is evaluated in this paper. Likewise, Eevee is a rasterizer renderer that scans the scene pixelwise and outputs it in real-time.

Additionally, the rendering tools were chosen because all of them provide a programming interface in the same language, that is Python. This was especially valuable for the implementation of the sequential rendering process discussed in section 3.5 because there was no need to drastically change the fundamental logic. Finally, except for Maya all software is freely available.

1.4 Choice of material parameters

Within the context of this paper, four material parameters will be examined in more detail. These are *Roughness*, *Specular*, *Metallic* and *Clearcoat*.
Roughness refers to the quality of the material surface. The rougher a surface, the more uneven is its texture and the more scattered the light is reflected. If the surface is not rough at all (i.e. ideal smooth), it represents an ideal mirror as light is reflected exactly into one direction.
Metallic is meant to be a binary parameter either set to 0 or 1. The parameter determines whether a surface appears metallic or non-metallic, or in other words: as conductor or dielectric surface. While dielectric surfaces normally are approximated in rendering with a diffuse and specular reflection term, metallic surfaces omit diffuse reflection (Guy & Agopian, 2020), as described in later sections.
Specular correlates with the intensity of the specular highlight. A lower value means that the specular highlight will be less prominent.
Clearcoat controls the visibility of a thin additional glossy layer on top of the surface of the object. It is often used to model materials like automotive paint.

One deciding factor for this choice was that all rendering tools rely on the metallic workflow (as explained in section 2.4.4). As Sturm et al. (2016) state, the "base of the metal-roughness material model consists of the [...] parameters BaseColor, Metallic [and] Roughness" (p. 120). Thus, *Roughness* and *Metallic* are crucial for defining a basic material in this workflow, which makes both parameters interesting for comparison. The other two parameters were chosen because both range from 0 to 1, however the implementation of the parameter *Specular* allows higher values than 1 in some rendering tools, as well. Also, the calculation of the effect behind the parameter *Specular* is rather simple compared to the computational costs of the more complex appearance feature *Clearcoat*. Both parameters are selected to provide different levels of complexity concerning the implementation of material appearance features. In terms of evaluating the quality of reproducible materials, especially the parameter *Clearcoat* was expected to provide some interesting information, as it is considered the most complex appearance feature in the chosen set. Altogether, the selected parameter set represents four well-known material properties for material design and should provide first insights to the intuitiveness of material editing and possible visual differences in the chosen rendering software.

2 Theoretical background and models for rendering materials

To understand why a certain colour materialises at the surface of an object in the final rendered image, it is important to understand how the underlying shading model of the renderer works. As Schlick (1994) encapsulates, the calculation of the material model is "the heart of every rendering method" (p. 1). In PBR pipelines, the material model is based on physical laws and often simplified by psychovisual insights about the human visual system. From these findings, mathematical models are derived and implemented in a concrete shading model. Finally, the parametrization of a shading model determines the scope for action in material design. Thus, it is important to consider all these levels when evaluating material models in current rendering software.

2.1 Material Appearance – Human perception of materials

The human perception plays a central role in the assessment of material appearance as it is important to recall that the perception of materials is not objective but rather subjectively shaped. Although this paper does not deal with this topic in detail, as the focus is rather on the implementation of material models than on the human perception of materials, this section is intended to make the reader aware of the influence of perception by giving an impression of the basic considerations for dealing with this issue.

Humans are able to draw conclusions about the properties of a material by perceiving its visual appearance. Fleming et al. (2003, 2015) found that humans memorise the optical properties of materials in connection with various lighting situations. If the object is viewed in a realistic lighting, humans can conduct other properties of the material with this knowledge, such as the age and value of the object consisting of this material. Furthermore, these findings suggest an inviting character or affordance for materials. The perceived material properties tell humans how to react and interact with objects. For example, material appearance partly tells us if an item is valuable and worth to collect, "whether the ground is safe to stand on or whether food is fit to eat" (Fleming et al., 2015, p. 1). The features that we attribute to an object depending on the impression of its material determine our opinion, expectations, and feelings about the item and hence impacts the assessment of material appearance in an individual, subjective manner. A key element in industrial design, for example, is "getting the *shitsukan* qualities of their products – their 'look and feel' – just right" (Fleming et al., 2015, p. 1). That is because the customer of a product will rate the product immediately and subconsciously just by viewing it in a commercial and associating different thoughts and emotions with the item depending on its look. Thus, a designer needs to precisely control the appearance of an object to evoke the exactly desired reactions to a product.

To be able to influence material appearance in an unambiguous way, it is important to establish a common terminology for the description of optical material features. For example, Jones (1922) distinguishes between "gloss" as being a physical or radiometric quantity and "glossiness" as the perceptual counterpart, "referring to the subjective sensation produced" (p. 146). Further, Hunter (1937) investigated the perception of gloss and formulated six different types of glossiness (p. 22). Two of those are still commonly used for wording the distinct features of gloss, which are *specular gloss* referring to the "brilliance of specularly reflected light [or] shininess" (p. 22) and *sheen*, meaning the "shininess at grazing angles" (p. 22). Meanwhile, further studies have been carried out to concisely describe also other characteristics of material appearance. For instance, Serrano et al. (2016) propose fourteen attributes about material appearance, including terms as *plastic-*, *rubber-*, *metallic-*, or *fabric-like*. Finally, there is no uniform terminology by now since the field has not been sufficiently studied yet and requires an interdisciplinary collaboration such as in the active EU research project "DyViTo" (2017). Eventually, the findings of the research on the human perception of materials are incorporated into the development of intuitive interfaces for material design.

2.2 Physical principles of reflection

To explain the resulting images of the physically based renderers, the most relevant physical principles used in the examined material models are outlined in this section. As formulas are not essential for understanding the basic principles, they will be explained in appendix A.2.1 for interested readers. Also, the insights on the physical perspective of light transport is not only important for the implementation of PBR material models but may be useful to understand other shading models as well. Especially procedural shading techniques tend to use some of the described principles as building blocks for calculating an artistic material appearance. For example, the Fresnel term is often used to add a vignette-like outline to a shaded mesh. With that, further visual effects can be created such as a ghostly look.

For all considerations, the *principle of energy conservation* must not be neglected. It states that the sum of the energies in an observed system is constant and energy can neither be created nor destroyed. Thus, when a light ray hits the surface of an object, the incident light energy must remain somewhere else. The further path of the light beam is determined by the material properties. The light ray can either be reflected, transmitted, or absorbed by the material. In this paper, only the light transport of reflection is analysed while transmission and absorption are both neglected. Please refer to appendix A.2.1 for further information on the law of refraction, or Snell's Law respectively, that deals with transmission. Generally, a distinction between specular and diffuse reflection is made (Hecht, 2018, pp. 205–206). As Hecht (2018) states, both situations are extremes, the behaviour of the most surfaces in the real world lies somewhere in between (p. 206). In general terms, the combination of both reflections is denominated as "glossy reflection" (Lensch, 2004, p. 20; Pharr et al., 2018, Chapter 8). Before discussing these reflections in more detail, one further principle will be explained. As Pfeiler (2017) notes, the so-called Helmholtz *reciprocity principle* states that the path of a light ray is reversible, which means that the position of the light source and the viewer or camera can be exchanged without any changes to the light transport except of the inverted direction. However, this principle is only valid if absorption and polarization changes of the light wave are neglected (pp.46-47) like in the context of this paper.

Specular reflection occurs at smooth material surfaces and causes the deflection of a light ray in exactly one direction. The path of the reflected light ray follows the law of reflection which states that angle of incidence is equal to the angle of reflection and the reflected beam has to lie in the plane of incidence (Zinth & Zinth, 2009, p. 32). For an ideal smooth surface, the incident light ray is not spread into several light beams. Instead, each incident light ray is deflected at the material surface into one direction. This corresponds to an ideal mirror because the reflected rays that reach our eye are the same light beams that were originally reflected by an object, resulting

in the same perceived image. For this reason, specular reflection is important for describing mirroring and shiny materials, such as smooth plastic or metals.

Diffuse reflection occurs when light hits a rough surface (Pfeiler, 2017, p. 234) and effects a scattering of the incident light ray in several directions. If a material is rough, its surface is uneven which can be approximated with the model of micro surface structure. This approach is crucial for the material model of Cook and Torrance (1982) and is hence discussed in more detail in section 2.4.2. For now, in simple terms the micro surface structure can be thought of as infinitesimal surfaces that act like ideal mirrors. That means that the law of reflection applies to diffuse reflection as well, but only in the domain of microstructures. The orientation of the microfacets deviates from the orientation of the macro surface depending on the roughness. If the material is ideal smooth, the microfacets align with the macro surface. The rougher the material, the greater is the number of micro surfaces that are arbitrarily rotated in other directions. So, when light hits a rough surface, it is scattered into many different directions, because one light ray hits several infinitesimal microfacets that reflect the incident light beam in vastly different directions. Hence, characteristically these materials tend to have no highlight and in an ideal form, their "reflection is view-independent" (Lensch, 2004, p. 20) as light is reflected equally in each direction. In literature, the behaviour of a purely diffuse reflecting surface is often approximated by Lambert's cosine law. However, as "this reflection model is not physically plausible" (Pharr et al., 2018, Chapter 8.3) and was empirically determined (Pfeiler, 2017, p. 234), this model will be discussed in section 2.4.1. According to Lensch (2004), "an almost perfectly diffuse material is chalk" (p. 20). Other real-world materials which mainly reflect diffusely are "matte paint" (Pharr et al., 2018, Chapter 8.3) or paper (Hecht, 2018, p. 206).

Finally, after defining the direction the Fresnel equations determine the intensity of the reflected light beam (Zinth & Zinth, 2009, p. 33). The amount of reflected light in relation to the amount of incident light is called *reflectance*. In summary, the Fresnel equations state that reflectance depends on both the incident angle of light and the *index of refraction* (IOR) of a material (Zinth & Zinth, 2009, p. 34). More precisely, "at grazing angles most of the light is reflected as if from a mirror" (Pharr et al., 2018, Chapter 8.2.3), which means that the reflectance is greatest when light hits the surface almost parallel at very shallow angles, often referred to as *grazing angles*. When observing an object, this effect generally occurs at the edges of an object which appear to be lighter than the inner areas. The steeper the light hits a material surface, the less light is reflected. If the light ray hits the surface perpendicularly, the resulting reflectance is minimal (Hecht, 2018, pp. 251, 271) and commonly referred to as F_0, meaning the Fresnel reflectance at an angle of incidence of 0°. Since the incident angle is known, this property can be converted into the IOR of a material and vice versa due to the correlation between reflectance and the refractive index. Thus, the Fresnel term is often defined by either the value of F_0 or the IOR. Moreover, a distinction is often made between *dielectrics* and *conductors*. Due to the laws of

optics, the conductivity of a material influences the reflective properties of a material. As light rays cannot penetrate noticeably into conductors (i.e. metals), transmitted light rays are absorbed almost immediately (Hecht, 2018, p. 266). As the IOR is linked to the effect of transmission, the IOR for conductors is given by a complex number with a real and imaginary part to consider this effect. For dielectrics (i.e. non-metals) the imaginary part is equal zero (Hecht, 2018, p. 270). In summary, the *Fresnel effect* can be observed on any material and occurs on surfaces that are hit by light at grazing angles. Illustrative examples are the reflections on water surfaces or the floor. The surfaces are very specular in the distance and become less reflective nearby as the angle of incidence of light decreases.

2.3 Mathematical description of material properties

In an abstract way, materials are diversions for light. When a light ray hits the surface of an object, its material determines which path the light ray will follow afterwards. Usually there are three options for redirecting light, as already discussed in the previous chapter: absorption, transmission, and reflection. Mathematical descriptions of materials are based on this view. However, it should be noted that the definition of the mathematical formula for material properties has nothing to do with whether the material model is physically plausible or not. Agreeing with T.-W. Schmidt et al. (2014), the following given examples for mathematically describing light transport at surfaces "relate primarily to physically-accurate light transport models and simulations, [but] non-physical or artistic models of local light interaction" (p. 6) are also comprised.

In the following considerations a local 3D coordinate system is assumed whose upward axis is determined by the surface normal. An illustration is given in Figure 1 and will be explained later. While it is also possible to describe materials in a global scene context, these approaches "constrain the possible viewer parameters or the freedom of choice of illumination" (Fuchs, 2008, p. 12). By observing the processes in the local coordinate space instead, light transport can be described regardless of any additional information about the light source or the viewer (i.e. camera in 3D rendering). Hence, in accordance with T.-W. Schmidt et al. (2014) "material interactions [are construed] as any local interaction that manipulates the distribution of light at a surface" (p. 6).

2.3.1 Assumptions on light transport

Local light transport at material surfaces is modelled in a function that describes the correlation between the incoming and outgoing light movement. This function is the basis for the shading model of a renderer. It plays a significant role in the rendering equation and thus partly determines which colour appears on the surface of a material. The dimensionality of the function

is dependent on how many quantities – usually physical ones – are considered to describe this motion. More details on the reduction of dimensionality will be given in the appendix A.2.2. For the following considerations it is assumed that light is transported without any time delay or change in wavelength. Furthermore, it is assumed that light may be deflected in a different direction when hitting the surface.

If light can travel underneath the material surface and leaves it at a different point from where it is incident, the function is called *bidirectional subsurface scattering distribution function* (BSSDF). If only reflection is considered, it is also denoted as the *bidirectional scattering-surface reflectance-distribution function* (BSSRDF) (Nicodemus et al., 1977, p. 4). Both model the effect of subsurface scattering that can be observed with materials like milk, marble, the human skin, or leaves. It emerges especially when one of the mentioned materials is pointed towards a light source. However, this effect will not be examined in this paper. Hence, a deflected light ray must leave at the same point where it is incident.

A material can have a certain texture which means that its appearance is locally altered. The resulting function is commonly denoted with the attribute *spatially varying* (Haindl & Filip, 2013, pp. 14–15; Hullin et al., 2013, p. 3). If the material is homogeneous and thus the material properties of an object are the same everywhere on its surface, this adjunct is missing. For simplicity, materials are assumed to be homogeneous for this study.

Often only one kind of light transport is represented in a function, for example the BSSRDF and the BRDF discussed below only considers reflection, while the *bidirectional transmission distribution function* (BTDF) just includes transmission. Nevertheless, there are also functions describing several light transport mechanisms in one function, such as the *bidirectional scattering distribution function* BSDF or the already mentioned BSSDF that consider both reflection and transmission.

2.3.2 BRDF

The *bidirectional reflectance distribution function* (BRDF) introduced by Nicodemus et al. (1977, pp. 5–6) is crucial for understanding modern material models, especially as every rendering tool examined in this paper is based on this idea. As mentioned before, this function only models the light transport of reflection. It is defined as the quotient of outgoing radiance L_o in direction ω_o and the incident irradiance[1] E_i from direction ω_i:

[1] Technically speaking, in contrast to radiance irradiance is not directional. However, following the ideas of gallickgunner (2018), it can be thought as integration of radiances over an infinite number of directions, like the upper or lower hemisphere. The differential irradiance describes an infinitesimal amount of this

$$f_r(\omega_i \rightarrow \omega_o) = \frac{\mathrm{d}L_o(\omega_o)}{\mathrm{dE_i}(\omega_i)} = \frac{\mathrm{d}L_o(\omega_o)}{\mathrm{dL_i}(\omega_i)\, cos\, \theta_i\, d\omega_i} \qquad 2.1$$

In simple terms, the function yields the amount of reflected light in relation to incident light (i.e. reflectance) for the rendering equation. It is called "bidirectional" as the reflectance is dependent on both, the incident and outgoing direction of light. Furthermore, the term "distribution" means that the function returns a reflectance value for any given position on the material surface. Hence, the BRDF "characterize[s] the reflectance properties of a point on a surface" (Adelson, 2001, p. 5) if it is spatially varying or the same reflectance value for any surface position in terms of homogenous materials, respectively. As light movement is considered in the local coordinate system of the material surface, the incident and outgoing direction is expressed by two angles θ and ϕ. Figure 1 illustrates the direction ω described in the local coordinate system of a material surface. The angle θ denotes the rotation away from the surface normal n and ϕ is the rotation angle around the up axis.

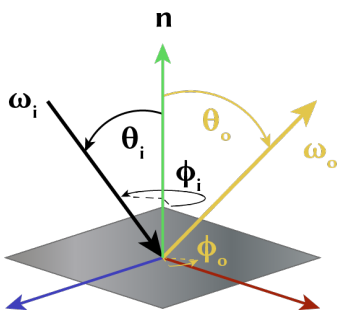

Figure 1: Local coordinate system for a material surface. The normal vector n of the surface is set as upward axis (green). Each direction ω (black for incident, yellow for outgoing ray) is described by two angles θ and ϕ.

Since there are two degrees of freedom for each direction, the BRDF is a 4D function. It describes light transport as deflection towards a certain direction

$$(\theta, \phi)_i \rightarrow (\theta, \phi)_o \qquad 2.2$$

and determines the intensity of the reflected light ray.

integration. Hence, the differential irradiance is equal to the integrand, i.e. the radiance in a certain direction, which is weighted with the cosine expression due to Lambert's cosine law.

2.3.3 Other light transport functions

Another function will be briefly outlined to give an idea of other possible forms of representations. The *bidirectional texture function* (BTF) is often used to represent measured data. It is probably less relevant for productions in the media industry since lighting conditions are almost predefined and static and requires high computational costs. Yet, this abstracted formulation of a material may become important for the industrial design sector, as it allows for a fast and accurate representation of materials on different objects. Haindl and Filip (2013) even claim that it is "the best [general reflectance function] approximation which can be managed with recent high-end technology and mathematical knowledge" (p. 21). The mathematical formulation assumes that the light source is infinitely far away of the illuminated object (Fuchs, 2008, p. 32). Thus, shadowing created by microgeometry no longer has an impact.

2.4 Material models in rendering pipelines

In the process of rendering an image the renderer must solve the so-called rendering equation to obtain the final image data. The renderer collects data about the lighting situation, scene geometry as well as material properties. The definition of material properties is drawn from the mathematical formulation of light transport, which is discussed in the previous section. It serves as abstract concept for the concretised material model of the renderer and indicates the dimensionality and thus the dependencies of the function for calculating light transport. By now, many approaches have been developed on the idea of a BRDF or BSDF, for example. The most important material models will be discussed only regarding the BRDF. Hence, only the effect of reflection is considered as before.

2.4.1 Phenomenological models

Phenomenological material models take the approach of transferring an observed appearance feature into controllable parameters. Sometimes, they are referred to as *empirical models* because this approach involves experimenting with the characteristics of the material properties rather than elaborating a method based on abstracted theoretical background knowledge like physical laws. Neither does this approach result in intuitive material design as it "does not give consideration to [...] meaning of individual parameters" (p. 128) as Haindl and Filip (2013) report. However, the implementations are often kept simple, which allows efficient computations. By now sometimes those models are also referred to as *traditional models* since the principled approach discussed in the next section is regarded as state-of-the-art. The three most popular models are briefly presented in this chapter.

First, there is the Lambertian material that is often used to describe an ideal diffuse surface, as adumbrated in section 2.2. Adelson (2001) describes a Lambertian surface as "an ideal matte surface that reflects light uniformly in all directions regardless of the angle of incidence" (p. 5). This behaviour is modelled with a BRDF that yields the same reflectance value for each incoming and outgoing direction of light. Thus, the Lambertian BRDF is a constant value (Haindl & Filip, 2013, p. 19). It derives from Lambert's cosine law and is defined as the diffuse colour, or albedo, ρ_0 normalized by π to ensure energy conservation (Lagarde, 2012a):

$$f_r(\omega_i \rightarrow \omega_o) = \frac{\rho_0}{\pi} \tag{2.3}$$

Secondly, Phong (1975) introduced a model that "should in some way imitate real physical shading situations" (p. 314). However, the model is not based on physical principles but rather attempts to reproduce the phenomenon of a specular highlight. Moreover, the Blinn-Phong model represents a variation of the Phong model. Both consist of three parameters for controlling the diffuse and specular colour (including intensity) of reflections as well as the size of the highlight. Nevertheless, as Haindl and Filip (2013) note, "it is very hard to find the relationship between the parameters […] and the physical characteristics of the represented material" (p. 129).

Lastly, the Lafortune model is also counted to the phenomenological models at this point, as it generalizes the Phong model by revising specific appearance features. Lafortune et al. (1997) themselves state that their model "can capture important phenomena such as off-specular reflection, increasing reflectance and retro-reflection" (p. 1). While basic physical principles are already fulfilled, including reciprocity, energy-conservation, retro-reflection and increasing reflectance at grazing angles (Lafortune et al., 1997), the model is not fully physically based as the starting point for the considerations were not physical laws such as the Fresnel equations. Furthermore, the influence of parameters remains unintuitive since they have non-linear effects on the appearance over a range of values that extends to infinity. Additionally, parameters are coupled in their effect on material appearance. For example, the four parameters controlling the shape of the specular highlight require time for experimenting with certain values as a modification of one parameter entails an adjustment of the other.

In conclusion, phenomenological models offer first approximations by reducing several optical phenomena to simplified formulas. While they are often used for "fast hardware implementation[s]" (Haindl & Filip, 2013, p. 128) due to the low computational costs, these models are no longer suitable for digital productions by now because vastly better qualitative results and improved workflows can be achieved by using principled models, which are discussed in the following section.

2.4.2 Principled models

In contrast to phenomenological models, principled models did not emerge from the motivation to empirically reproduce certain material appearance features. Rather, the goal is to analytically define abstract principles that exceed the mere reproduction of material appearance in a first step. Subsequently, a greater scope for design can be guaranteed because the model is decoupled from other dependencies like lighting and abstract concepts are often more accessible. In other words, a well-designed material model should provide added value for digital productions, like Burley (2018), for example, says: "Our motivation for using physically based rendering is primarily artist productivity rather than any explicit goal of increased realism" (p. 2). The two most important principles discussed below are firstly the orienting on physical laws to ensure continuity for materials in different lighting situations, and secondly the improvement of the material design process by a user-friendly parametrization of the underlying calculations. In current rendering pipelines both principles are usually adapted, with each renderer using its individual implementation and assumptions.

Physically based models

The term "first principles models" was used by Dorsey et al. (2007, pp. 83–84) to describe material models that are based on physical principles and consider material properties such as the IOR. However, physical accuracy is compromised by computational costs, so these models do not claim physical correctness. The most important superior principles to consider are "energy conservation, reciprocity rule [and the] microfacet theory" (Schlick, 1994, p. 1), which were already explained in section 2.2. Probably the best-known material model among physically based ones is that of Cook and Torrance (1982). The base concept is commonly adapted in modern rendering pipelines, just like in the four chosen tools for this thesis (cf. Blender Foundation, 2020b, 2020c; Epic Games, 2020c; Georgiev et al., 2019b). Several authors including Ward (1992) and Schlick (1994) advanced the model by extending its functionality to reproduce a wider range in materials, such as anisotropic materials, and lowering computational costs by making simplified assumptions. However, the fundamental idea remained the same. The model considers three basic elements for computing the reflectance of a material: the Fresnel term F, the shadowing-masking term G and the microfacet distribution D. Again, the formula is not presented here for simplicity, but will be given in the appendix A.2.3.

First, the Fresnel term F is based on the Fresnel equations discussed in section 2.2. In current rendering pipelines, the approximation by Schlick (1994) is widely adopted. It represents a simplification of the previous one used by Cook and Torrance (1982) that "can be computed almost 32 times faster with less than 1% error" (Schlick, 1994, pp. 7–8). A distinction is often

made between conductive and dielectric materials to further simplify implementations, as dielectrics do not have a complex IOR like conductors (see section 2.2).

The other two elements included in the Cook-Torrance model are based on microfacet theory, which has already been touched in section 2.2. The main idea builds on the reason for diffuse reflection in reality. Real-world objects appear to be rough if their surface is not perfectly smooth, but rather uneven as if the surface consists of many small micro surfaces. These are called *microfacets* and in most material models[2] they act as infinitesimal mirrors that reflect incident light specularly (Dorsey et al., 2007, pp. 86–87).

The microfacet distribution D is a function that indicates the statistical distribution and orientation of these micro surfaces. As the orientation of a surface is defined by its normal vector, it is also referred to as "normal distribution function" (Heitz, 2018, p. 1). By now, there are several approaches to describe this microscopic texture of a surface. The most known distribution functions include Beckmann, GGX introduced by Walter et al. (2007) and the Phong distribution (Walter et al., 2007, p. 6). An extensive discussion of each would be beyond the scope of this paper. However, it should be noted that the GGX distribution is currently widely adopted in rendering pipelines, for example in all examined renderers in this paper according to their source code (cf. Blender Foundation, 2020b, 2020c; Epic Games, 2020c; Georgiev et al., 2019b).

Lastly, the shadowing-masking term G covers the effect that some microfacets will not receive any incident light because it is blocked by other micro surfaces. As this phenomenon is highly dependent on the geometrical structure of the microfacets, this term is coupled to the distribution D discussed before. A well-known approximation for this term was developed by Smith (1967) and is used in UE4, for example (Epic Games, 2020c).

In summary, it can be stated that physically based material models analytically observe the mechanism of light transport at material surfaces. Because a lot of research has already been done in the physical field of optics, the models make use of these laws. As a result, even photorealistic results can be achieved in physically based rendering pipelines. Since the material properties are abstracted and not bound to a specific material appearance feature, materials can be viewed flexibly in different light situations.

[2] As Dorsey et al. (2007) state, this is, for example, not the case with the Oren-Nayar model, as "the microfacet reflectance […] is assumed to be Lambertian" (pp. 86.87).

Intuitive models for creative workflows

Later, Burley (2012) expanded the term "principled models" to consider the intuitiveness of material design as a new field for material model principles and to distance his proposed model from strictly physically correct ones. The central issue for intuitive and "artist-friendly" (Gulbrandsen, 2014, pp. 64–65) material design is the parametrization of the material model. As in the case of physically motivated models, overall principles are first established, which must be achieved when developing the concretized material model. Hence, Burley (2012) formulates the following five principles for intuitive material design: The size of the parameter set should be minimal, parameter names should be "intuitive rather than physical" (p. 12), range "from zero to one over their plausible range" (p. 12), while at the same time may "be pushed beyond their plausible range where it makes sense" (p. 12) and all design combinations should be "as robust and plausible as possible" (p. 12), which means that any combination of values should generate a believable material. In collaboration with artists, he has designed a new material model that meets artistic demands as well as basic physical assumptions. Karis (2013) adapted the model for the UE4 and extended it by further principles that are important in the context of real-time graphics, such as efficiency with many simultaneously visible lights and the use of only one base shading model for their deferred shading pipeline (pp. 1-2). Moreover, he emphasizes the need to "enable non-photorealistic rendering" (p. 2) as well, although this was already relevant for Disney's shading model presented by Burley (2012) but not yet included as a principle. In summary, principled models for intuitive material design partly depend on the use case in production, but also rely on the human perception of material appearance for intuitive descriptions of material parameters and their influences.

Today, most of the modern rendering tools have adopted these suggested principles in their PBR pipeline and enabled the use of "a single BRDF on everything" (Burley, 2012, p. 18). This definitely simplified the workflow for creating physically plausible scene setups, as Burley (2012) already had endorsed. The incipient process of attuning the material models among rendering software had also established material libraries and tools dedicated to material design, such as the application suite Substance by Adobe.

Meanwhile, two commonly used workflows have evolved in terms of material design for PBR, which are referred to as *specular* and *metallic workflow*. Both differ in the encoding of the reflectance properties and colour for metallic materials. On the one hand, the specular workflow clearly distinguishes between diffuse and specular reflection, represented by two 3D colour inputs, often called *Diffuse*, or *BaseColor*, and *Specular*. As metals do not have a diffuse reflective component as explained in section 2.2, the *Diffuse* input "pure black (0.0) indicates raw metal" (Sturm et al., 2016, p. 120). Thus, the perceived colour of metals results entirely from the colour of the *Specular* input, while the colour of dielectric (i.e. non-metallic) materials is specified

by both *Diffuse* and *Specular* inputs individually. For example, this means that the colour of the specular highlight can be defined directly with the *Specular* value. As opposed to this, the metallic workflow first declares if a material is metallic or dielectric using the 1D *Metallic* input and then treats the 3D *BaseColor* input either as a diffuse and specular component for non-metallic or specular only component for metallic materials. This workflow is "more memory friendly" (Sturm et al., 2016, p. 121) with the cost of no possibility to set an individual highlight colour or intensity (i.e. Fresnel reflectance). To still be able to set at least a custom intensity for specular highlights, many models have an additional 1D *Specular* parameter. Although the specular reflected colour cannot be set, this can also be an advantage as in physical terms only metals can have tinted highlights. Additionally, the physical principle of energy conservation is violated less easily. An ideal smooth dielectric material thus cannot reflect more light than is incident.
Both workflows have advantages and drawbacks, although at present the metallic workflow is used predominantly in rendering software, for example, in UE4, Blender or the Arnold renderer. However, some tools still support both workflows, like for example the universal rendering pipeline of the game engine Unity or the texture software Substance Painter by Adobe.

2.5 Material design

The subject material design deals with the process of editing the material appearance to match a desired look or art style. In digital productions, the intuitiveness of material design is influenced by several factors. The editing process through an artist can be compared to any other user interaction task. The user of the application is represented by the artist, whereas the application itself corresponds to the respective rendering software. Schmidt et al. (2014) even claim that "artistic editing should never be isolated from user interaction" (p. 4). Thus, the authors maintain the "three interaction paradigms" (p. 4) defined by Kerr and Pellacini (2009, 2010). These consist of "direct interfaces", "indirect interfaces" and "goal-based interfaces" (Kerr & Pellacini, 2009, 2010, as cited in T.-W. Schmidt et al., 2014, p. 4). In terms of material design, the mostly used method of interaction in current digital production workflows are direct interfaces. Straightforwardly, the user can edit the appearance of a material by varying the parameter values of the underlying material model. As the other two forms of interaction are not implemented in the chosen rendering tools yet, the indirect and goal-based interface for material editing both are not considered in this paper and are neglected at this point.

As Schmidt et al. (2014) endorse, direct interfaces are often perceived as "unintuitive as [the] parameters often expose too many degrees of freedom to a user" (p. 7) and frequently it remains unclear which appearance feature is controlled by a specific parameter. For example, the visual impact of the parameter *Clearcoat* on the material appearance strongly varies depending on the

lighting scenario, as illustrated in section 4.2.4 in detail. Thus, the probably expected effect of the parameter *Clearcoat*, a thin additional glossy layer on top of the surface of the object, is only or best observed under certain lighting conditions. Hence, during editing material parameters, the user sometimes must guess the meaning of the parameter *Clearcoat* since the direct visual feedback might not be accurate. In fact, in some lighting situations the influence of *Clearcoat* might even be mistaken for *Roughness* or other parameters if the user is only guided by the visual feedback of the viewport and does not consider the meaning of the parameter[3]. Altogether, the way of editing materials via a direct interface might be simple to understand, but influencing the final appearance is not always intuitive (cf. T.-W. Schmidt et al., 2014, p. 7).

Eventually, the intuitiveness of material design is heavily dependent on the description of the parameters. Especially in the PBR workflow, the parametric material model often contains complex physical terms internally. The challenge in making the design process user-friendly is to create an easily understandable interface which is adapted to the human perception of material appearance. The intuitiveness of material editing in PBR pipelines depends on how well the material appearance is described by a given parameter set and how those attributes are translated into physical material properties.

3 Methodology

This chapter states how the material models of different rendering tools will be evaluated. First, the basic strategy for getting results and evaluating them is illustrated. Afterwards, a suitable test environment is defined and implemented. At the end of this section, noticeable issues regarding the implementation of the sequential rendering process are portrayed.

3.1 Evaluation strategy

To investigate how the PBR material models of the chosen rendering tools differ in terms of the quality and range of reproducible materials, several images depicting an object with different material properties in a certain lighting condition will be generated. Guarnera et al. (2018) refer

[3] This raises the question of how well the PBR workflow is understood by artists and whether there is a correct or incorrect way to design materials. An in-depth investigation of this issue is beyond the scope of this paper. However, it should be noted that Hiranyachattada and Kusirirat (2020) dealt with the first question. The authors aim to improve the understanding of material parameters using an augmented reality application. By an inquiry by e-mail, which is given in appendix B, I found out that the authors used parameter values from the Unreal Engine 4 documentation as ideal and "correctly set" values. But the latter question remains: Is there a right and wrong setting of the parameters for PBR materials? How much creative freedom remains for the artist?

to the resulting images as “BRDF Difference Probe[s]” (p. 1, 5). The material property is influenced by exactly one parameter at a time. Each time the value of the parameter is incremented with a specific step size, a new image is created. This results in a row of images for each rendering tool, which are contrasted in a table, also referred to as material chart. The layout is inspired by the illustrations of Burley (2012, p. 13), several examples of Lagarde (2011b, 2012b, 2014) and Unity Technologies (2014).

Within the context of this paper, four material parameters will be examined in more detail, as already explained in section 1.4. The used shader models are the “Principled BSDF” in Blender, the “Standard Surface” shader in Arnold and the “Default Lit” or “Clear Coat” model, to evaluate *Clearcoat* respectively, in the Unreal Engine. In some cases, the names of the parameters vary slightly between each rendering software, for example Arnold distinguishes between *specularRoughness* and *diffuseRoughness*. In this case, the parameter with the more similar visual impact is selected. In the above example, *specularRoughness* behaves analogous to *Roughness* in the other rendering tools as both control the sharpness of the specular highlight. Also, *Metalness* in Arnold was used to evaluate the parameter *Metallic* in this paper, as well as *Coat* corresponds to *Clearcoat*.

Since a comprehensive quantitative investigation was beyond the scope of this paper, a qualitative comparison method is chosen. The images will be rendered at a resolution of 500x500 pixels and then compared in three ways: by analysing and opposing the unchanged output images, by looking at the difference images between each parameter alteration step and by studying selected scanlines of an image. The plots and difference images for the latter two methods were computed with JupyterLab and the implementation is given in the appendix A.1.

3.2 Defining the test environment

To generate images showing the material properties and appearance of an object, a scene environment must be defined first. According to Schmidt et al. there are “several controllable appearance parameters” (2014, p. 2), in particular “the position, orientation, and emission profiles of light sources [...]; the camera parameters [...]; the materials [...] of each object; the light transport simulation algorithm and its settings”. As the latter differs in each rendering tool, some visual differences will be inevitable. The implementation of the light transport will not be manipulated, because this would not emulate a normal working environment of a 3D artist. However, this paper is dedicated to study visual differences in a regular digital production workflow.

Still, the other three parameters, camera, light, and geometry, can be adjusted rather precise. Ideally, those three items are the only factors with an impact to the result. This means that other

issues such as post processing and colour management should either be set exactly in the same way or turned-off if possible. This way the test environment should cause the same output image in each renderer, aside from artifacts arising from different light transport algorithms and settings. Since this is inevitable, the lighting settings will be discussed as a last point.

3.2.1 Camera

The camera is responsible for setting the framing of an image. This is influenced by the camera position, focal length, and field of view and the sensor size, respectively. Also, the aperture affects the look of the resulting image, but not its framing. In fact, "lens model [and] shutter time" (T.-W. Schmidt et al., 2014, p. 2) influence the image appearance as well. This is neglected at this point since the user cannot change these parameters in neither of the chosen rendering software.

The distance of the camera to the sphere is set to 10m, while the focal length is set to 170mm. This was found to rule out perspective distortion effects. The sensor size is set to 36mm for width and height, resulting in a quadratic image.

3.2.2 Lighting scenario

The lighting setting is crucial for the perception of materials. Fleming et al. (2003) observed that humans perceive material appearance as the interplay of lighting conditions and material properties. Some effects like subsurface scattering or the Fresnel effect can only be observed if the object is illuminated in a special way (Bousseau et al., 2011). Moreover, Fleming et al. (2003) state that human-beings use statistical information they have learnt about different lighting scenarios for drawing conclusions about the properties of a material. As the authors prove by a matching task experiment with four participating subjects, humans can realize surface reflectance better if the lighting conditions are natural and close to the real world.

For this paper, two fundamentally different lighting scenarios were chosen. The first one consists of a natural lighting situation, which is implemented using an environment map as explained below. This setting addresses the natural human perception of material appearance while at the same time it is an often-used technique in digital production workflows, for example in architecture design or virtual film production. In contrast, the second lighting scenario is a very unnatural one, composed of only one directional light. This illumination is used to provide images that are generated with a more basic lighting feature. Besides, the different implementations of image-based lighting in each rendering tool must be considered as well as they may cause a different output image. The lighting scenario with one directional light is more likely to be implemented in the same manner in each software than environment maps, because the latter strongly depend on several factors such as the colour management of the texture inside

the engine, the generated mipmaps, texture filtering, and the used method for unwrapping the texture. In contrast, directional lights are commonly implemented in such way, that the user can control them with two parameters, the power of the light source and the rotation or direction of light, respectively. So, this lighting scenario may be less error-prone as there are less causes for mismatch.

Environment maps are textures, which act like a background and are used for image-based lighting. Often, rendering tools visualize this light source inside the editor by mapping the texture onto a sphere that surrounds the scene (Pharr et al., 2018, Chapter 12.6). They can be thought of as "infinite area lights" (Pharr et al., 2018, Chapter 12.6), that illuminate the scene from an infinite distance with the colour and intensity of each texel of the underlying high dynamic range image. Thus, the illuminated object looks like it was situated in that environment. Four different environment maps were chosen to evaluate different material appearance features. All high dynamic range images (HDRi) are freely available and were taken from the website HDRIHaven (Zaal, n.d.). Please refer to the appendix A.1 for the image data. The chosen lighting scenarios range from high to medium dynamic range, as depicted in Table 1. Initially I thought that including a dark illumination scenario like the night scene "Moonless Golf" would provide added value for recognizing the interplay of lighting and material appearance. However, a closer inspection of my findings revealed that this is not likely to be the case. Nevertheless, the environment map was still valuable for determining sufficient lighting settings, as discussed in section 3.2.5, and the results of the renderings in this environment will be included in the appendix A.1, as well.

	Colorful Studio	**Lebombo**	**Moonless Golf**	**Sunflowers**
Dynamic Range	very high	medium	extremely high	extremely high
Exposure Values	15	9	23	23
Whitebalance	3700	4200	3942	6550

Table 1: Properties of used environment maps

3.2.3 Geometry

The choice for the illuminated object depends on which material characteristics should be observed. The shape of an object can either be concave or convex. For simplicity, a convex sphere will be used to observe the differences in material appearance between the chosen rendering tools. This has the advantage that important appearance features like the Fresnel effect can be observed towards the peripheral areas of the sphere. Also, the camera positioning is completely independent of the geometry of the object, as the 2D projection of a sphere appears to be a circle from any desired viewing angle. Other geometrical shapes are not examined in this paper but may provide additional information about material appearances features as well.

Besides the shape of an object, the resolution of a surface mesh is critical as well. Within the chosen rendering platforms, geometry is built up from small triangles or quads into a polygonal mesh. The resolution of the polygonal mesh, that is how many triangles or quads are used within a certain mesh area, changes the appearance of the output image. Low poly meshes have a significant lower polygon count compared to high poly meshes. A too low resolution of the polygonal mesh may result in artifacts such as the so-called *terminator problem*. This issue occurs when rendering a low-resolution polygonal mesh illuminated by hard light that casts distinct shadows. As Woo et al. (1996) state, this artifact happens due to "improper self-shadowing" (p. 22). It becomes visible as step-like dark blocks in the transition area between the shadow and illuminated surface of an object, also known as terminator. Figure 2 depicts the artifact occurring with the lower-resolution mesh on the left and the smoothed (i.e. subdivided) mesh on the right.

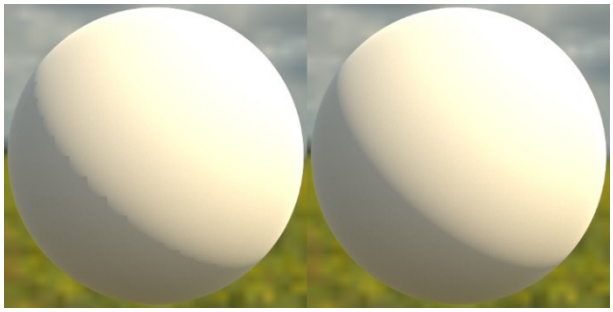

Figure 2: Terminator problem in Cycles

For the test scene, a polygonal sphere with a vertical and horizontal resolution of 50 subdivisions each is used. But, in Maya and Blender it is possible to make use of the Catmull-Clark algorithm when rendering the scene. The surface is subdivided internally before the rendering process starts. Since the terminator problem only occurred in Cycles with a resolution of 50x50 subdivisions for the sphere, I applied two further subdivisions with the Catmull-Clark algorithm, which appears to be the default setting in Arnold, or Maya, respectively (Solid Angle S.L., n.d.-c). Also, I exported a three times smoothed sphere as FBX file from Maya and imported the mesh into UE4. There was a noticeable difference when exchanging the mesh in the scene with a directional light. However, I did not swap the geometry for the scene with the environment map, as visual differences were not detectable, and those changes were made at a late stage already.

3.2.4 Default material

As only one parameter will be varied at a time, all other parameters stay fixed at a given value. Those values are set so the values depicted in Table 2. All parameters not listed in the table are set to 0, so that they should have no impact to the result. These are for example *Subsurface*, *Sheen*, *Anisotropic*, *Transmission* or *Emission*. The influence of these parameters will not be covered in the scope of this paper.

The parameter *BaseColor* (i.e. the albedo color) is set to (0.8, 0.8, 0.8) for all renderings. Without going into further detail, this value corresponds to the albedo value of fresh snow (Coakley, 2003, p. 1920) and represent a high, but still reasonable albedo value from a physical point of view (cf. North Carolina Climate Office, n.d.). Values beyond that this are rare in nature and thus not physically plausible.

The value of *Roughness* or *specularRoughness* for Arnold, respectively, is set to 0.2 per default for the purpose of the studies. This corresponds to a slightly rough material, which is not an ideal mirror, but also not so rough that the light is scattered in all directions. Hence, material features like specular highlights or the smoothing influence of clearcoat can be observed when examining the parameters *Specular* or *Clearcoat*, respectively.

The origin of the default *Specular* value lies in the calculation of the IOR for the rendering software Blender and UE4. While the Arnold renderer specifies the IOR as separate parameter to control the "balance between reflections on surfaces facing the viewer and on surface edges" (Solid Angle S.L., n.d.-a), that is F_0 and the reflectance at grazing angles, the other three renderers take a different approach. The IOR used for the calculation of the Fresnel term in Cycles, Eevee and Unreal is encoded in the *Specular* value. Depending on whether the material is metallic or not, the value for F_0 – the reflectance value at normal incidence – is derived either from the *BaseColor* for metals or from the *Specular* value for dielectrics (see section 3.3 for UE4 source code). As the value for F_0 is correlated to the IOR, both values are determined by the *Specular* value for dielectric materials or the *BaseColor* for conductors. For the *Specular* value of the default material this means that a value of 0.5 corresponds to an IOR of 1.5 or a reflectance of 4% at normal incidence, respectively. This value approximately corresponds to the IOR of glass (Zinth & Zinth, 2009, p. 34).

Lastly, the values of *Clearcoat* and *Metallic* were both set to 0.0 to exclude effects by those two parameters when examining the impact of *Roughness* or *Specular*. This also means that only dielectric (i.e. non-metallic) materials will be examined when observing the influence of Roughness and Specular in section 4.

	Arnold	**Cycles**	**Eevee**	**Unreal**
BaseColor	(1.0, 1.0, 1.0) * 0.8	(0.8, 0.8, 0.8)	(0.8, 0.8, 0.8)	(0.8, 0.8, 0.8)
Roughness	-	0.2	0.2	0.2
Specular Roughness	0.2	-	-	-
Specular	0.5	0.5	0.5	0.5
Metallic	0.0	0.0	0.0	0.0
Clearcoat	0.0	0.0	0.0	0.0
IOR for reflection	1.5	-	-	-

Table 2: Properties of default material

3.2.5 Lighting settings

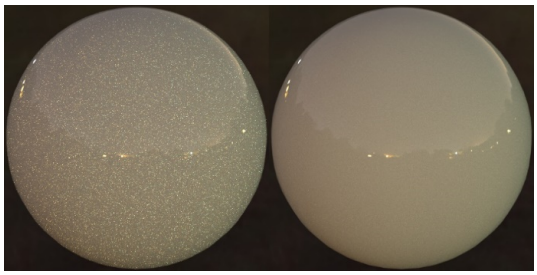

Figure 3: Fireflies artifact in Cycles

As already mentioned, visual differences between the rendering tools will be inevitable because of different lighting transport algorithms. The underlying calculations happening inside path tracing algorithms and rasterizer engines result in a significantly different appearance when analysed in detail. The goal is to minimize the visual differences that arise out of different rendering algorithms to ensure comparability among the different rendering tools. Thus, each rendering setting is set to output an image with the highest possible quality to minimize rendering artifacts. The used lighting settings for the results in this paper are listed in appendix A.3.

In terms of rasterizers, the number of samples plays a secondary role, as artifacts like noise or fireflies are not immediately perceptible due to interpolation of the calculated pixel values. Hence, the influence of lighting settings is not discussed further for the rendering tools Eevee and UE4. However, for both path tracing renderers Arnold and Cycles, it is crucial to render the scene with enough samples for lighting. The number of samples determines how often a light ray can bounce of a surface in the scene. In Cycles, the method "Branched Path Tracing" was chosen to have more control over the sampling settings and to set similar settings as in the Arnold renderer. Hence, both rendering tools allow to set a different number of samples per material feature.

Moreover, in Blender there is the option to set the number of samples used for sampling the environment map. If the number of samples is insufficient, rendering artifacts like noise or *fireflies* can occur, as depicted in Figure 3. The randomly spread light pixels that look like glitter or glowing fireflies, hence the name, come from the accidental sampling of an extreme bright light source, that is not weighted with the other scene lighting because there are not enough samples for comparison. To determine proper lighting settings, several test renderings were performed. It was noticeable that the resulting images of Cycles had more noise than Arnold with the same settings. However, for reasons of time this is not to be further investigated in the scope of this paper. Instead, many different settings with increased samples for different material features and a higher allowed number for light paths were tried out, which are given in the appendix A.3 as well. Eventually, I could choose between a good compromise of render time and

quality or a setting with exceedingly long render times but a slightly better quality. I decided to use the latter one for this work, as rendering times played a secondly role for assessing the representation of the materials in the rendering tools.

3.2.6 Colour management

Two important colour management issues are considered in the methodology. The first concerns the colour management of the environment maps. Test renderings have shown that the same texture interpolation method cannot be used in each rendering tool. Eevee for example only supports the texture interpolation method "Closest" according to various online forum entries (*Mipmaps or Nearest Filtering Option*, 2018). If the texture interpolation method is set to "Cubic" anyway, the artifact illustrated in Figure 4 occurred when using the environment map "Sunflowers". While same material is assigned to all objects, the sphere in the middle appears to be more reddish compared to the other two spheres. Interestingly, the artefact only emerged in this particular lighting situation.

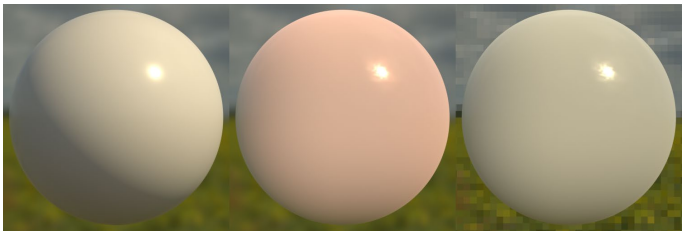

Figure 4: Effect of different texture interpolation methods. From left to right: "Cubic" in Cycles, "Cubic" in Eevee, "Closest" in Eevee.

The other issue is the output transform of the rendered image. All images will be rendered in linear space to ensure linear light transport but compared in the sRGB colour space, as this corresponds to the way that an artist generally views the editor when designing materials. Also, the image files are stored in the PNG format since it supports lossless data compression. To ensure that postprocessing does not affect the look of the image, a neutral tone mapper was used for all render engines.

3.3 Matching unit systems

Another important issue to consider is that each rendering software may use different units for lighting and position declaration. The position of an object is defined by its translation, rotation, and scaling. As the scale factor will always be equal one, scaling will be disregarded in the current considerations. The different unit systems for the chosen rendering tools and additionally the game engine Unity, for illustrative reasons, are compared in Table 3. It shows the default units for the respective item, as well as all possible selectable alternative units.

	Unreal 4.25	Blender 2.83.1	Maya 2020	Unity 2019.3.14 (without HDRP)
Translation units	cm (default), m	m (default)	cm (default), m	m (default)
Up axis	Z	Z	Y (default), Z	Y
Forward axis	Y	-Y	Z	X
Light units	"intensity", lx, cd/m², cd, lm	Watt, "strength"	"intensity"	"intensity"

Table 3: Different unit systems for commonly used rendering software. The terms in inverted commas are unitless quantities.

Yet, the unit system should be the same to ensure comparability. To face this issue, several cross-rendering pipeline tests were performed. The biggest issue was the rotation of the environment map in UE4. Unfortunately, I was not able to set the exact same framing for the environment map in UE4 as in other rendering tools. This is because Unreal provides its own implementation for the so-called "HDRI Backdrop", which is not an environment map in the manner of Blender or Maya because shadows are casted as well. This feature is intended to "showcase a model in a visually rich context with minimal effort" (Epic Games, n.d.). Although the implementation is accessible as an Unreal Blueprint, I was not able to determine the relation between the rotation of the environment map in UE4 and the other rendering tools. Consequently, the orientation of the light source slightly deviates from other rendering software, which in turn impacts the material appearance because of the dependence of the BRDF on the angle of incidence of light. Hence, the data of the UE4 renderings need to be interpreted with caution in direct comparison with the results of the other rendering tools.

To ensure that the brightness in all scenes is coordinated across the chosen renderers, the lighting scenario with the directional light was examined regarding the average brightness of the resulting image. For this investigation, a Lambertian material was used, as it was presumed to appear the same in each rendering tool due to its rather simple implementation. Surprisingly, I did not find any other solution to define a Lambertian material inside the UE4 than editing the source code of the engine itself. Because the effort for this would be unreasonably high, a material based on the shading model "Default Lit" was used instead. Its parameters were set to the following values:

```
BaseColor = (0.8,0.8,0.8)
Metallic  = 0
Specular  = 0
Roughness = 1
```

The *BaseColor* matches the albedo that is used for creating the material charts. Also, the parameter *Roughness* is straightforwardly set to its maximum value 1 to approximate the Lambertian material, that is an ideal rough surface, for cross-renderer comparisons. For the parameters *Metallic* and *Specular*, I referred to the source code of the BRDF of the "Default Lit"

shading model, retrieved from the GitHub website of Epic Games (Epic Games, 2020c). In "ShadingCommon.ush" the value of F_0, which is the Fresnel reflectance for the incident angle of zero degree as explained in section 2.2, is calculated from the values for *Specular*, *BaseColor* and *Metallic* (ll. 94 – 97):

```
float3 ComputeF0(float Specular, float3 BaseColor, float
Metallic)
{
      return lerp(DielectricSpecularToF0(Specular).xxx,
BaseColor, Metallic.xxx);
}
```

Depending on whether the material is metallic or dielectric, F_0 results either from the value for *BaseColor*, which is the albedo of a material, or the parameter *Specular*, which is multiplied by the factor 0.08 in the method "DielecticSpecularToF0(float)". That means for dielectric materials, a *Specular* value of 0.5 means 4% reflectance if light hits the surface perpendicularly. This corresponds to an IOR of 1.5 and occurs for example between air and glass surfaces (Zinth & Zinth, 2009, p. 34). While real-world materials can be either purely metallic or only dielectric, the parameter *Metallic* in UE4 allows for intermediate values probably for artistic reasons. As Burley (2012) states, artists "will need to interpolate between all of the parameters" (p. 16). The intermediate value of *Metallic* is used for the linear interpolation of the two methods for calculating F_0. For instance, if a material has a *Metallic* value of 0.75, the Fresnel reflectance for perpendicular incident rays is calculated as three quarters of *BaseColor* and one quarter of the influence of the *Specular* value.
Now, to create a nearly Lambertian material in UE4, it is also important to know that the F_0 term only defines the specular Fresnel reflectance. In the code line 994 of the "BasePassPixelShader.usf" it says that the *SpecularColor* is set to the calculated F_0 term like explained above. However, the *DiffuseColor*, which is the only term to consider for a Lambertian material, is calculated like this (l. 1023):

```
GBuffer.DiffuseColor = BaseColor - BaseColor * Metallic
```

So, to ensure that the *DiffuseColor* is only determined by the *BaseColor* as well as to extinguish all specular reflections for the quasi Lambertian material, first the parameter *Metallic* is set to 0. Since the resultant material is dielectric, the specular Fresnel reflectance at zero degree is calculated entirely by the *Specular* value. If this parameter is set to 0 as well, the F_0 value and thus the *SpecularColor* of the material are equal 0, too. Therefore, no specular reflection occurs at the material surface and it is a proper approximation to the Lambertian materials of the other rendering tools.

Each scene consisted of a sphere illuminated by a directional light source whose intensity was normalized to 1 in the respective unit of the rendering software. As modern rendering tools compute lighting in linear space, the images were output linearized for comparison to get a snapshot before gamma correction is applied. This way it was also possible to calculate a scaling factor that has a linear effect on the images. Hence, potential differences in brightness can be compensated by multiplying the original light source intensity with the computed scaling factor. The scaling factor was calculated as the ratio of two average image intensities, whereby the Arnold renderer was used as the comparative image (see appendix **Error! Reference source not found.**). Figure 5 depicts the results of the analysis with the rendered images on the one and the calculated values on the other side. As expected, the rendering from UE4 behaves differently than the others, presumably due to the different shading model. It is noticeable that the average overall brightness is significantly lower compared to the other rendering tools. To match the brightness of the Arnold renderer, the intensity of the directional light should be increased by about 5.6% in UE4 for the present case. However, these findings need to be interpreted with caution, since both material models are different. Secondly, it is interesting that Cycles and Eevee return different values for the average brightness of the total image. Both renderers rely on the exact same scene setup, material node and light intensity. Nevertheless, the difference is minuscule and may result because of the different underlying light transport algorithms. Thus, it will not be considered further at this point. To match the light intensity of Blender (i.e. Cycles and Eevee) with Arnold, the current intensity must be decreased by about 1%.

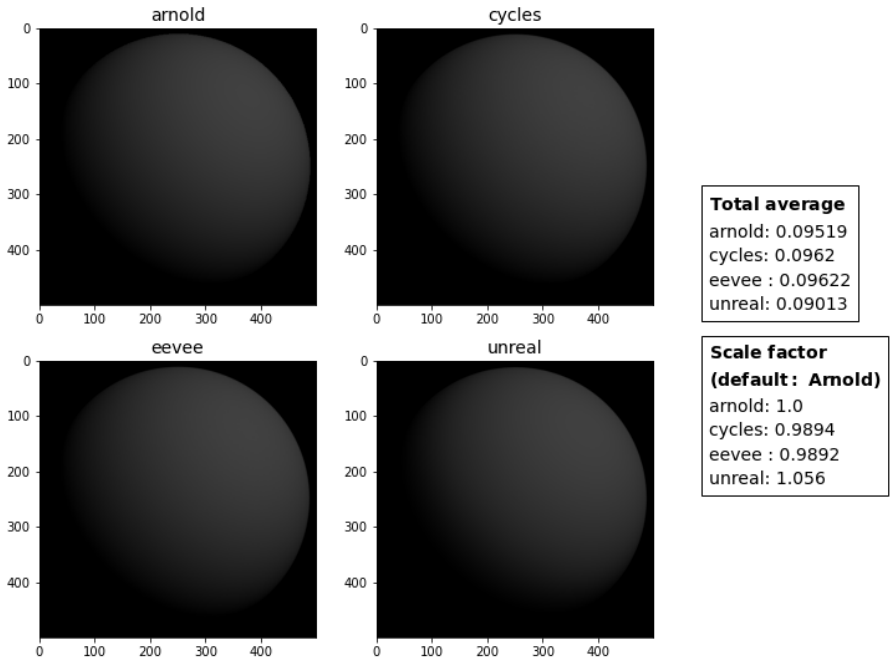

Figure 5: Brightness adjustment with Lambertian materials illuminated by a directional light

Altogether, the results are understood as negligible error for matching the light units of the directional light source across all renderers. Regrettably, for reasons of time I was not able to repeat the same process for matching the brightness of the environment maps across the rendering tools. Hence, the rendered material chart with environment maps cannot be compared among renderers as easily as if the overall brightness were matched. Methods like computing difference images will consequently be only applied within one rendering tool.

3.4 Implementation of the test environment

To ensure that each rendering tool arranges the object in the scene in the same way, a basic scene was created in one 3D software and exported as an FBX file afterwards. The FBX file format had "originally [been] developed by Kaydara for MotionBuilder, [then was] acquired by Autodesk Inc in 2006" (Blender Foundation, 2013). Today, it is a popular and widely supported format for exchanging 3D data, such as 3D models, light, camera and mesh data, animations or even materials (cf. Montagne, 2018).

For creating the test environment, I placed the previously described sphere at the scene origin, set its radius to 1 meter and positioned the camera 10 meters in front of the viewed object along the positive forward axis. The original scene was created in the rendering software Maya. However, the orientation of the subsequently added environment map is attuned to Blender's default orientation at 0°, which corresponds to a rotation of 90° around the y-axis in Maya. This decision was made to minimize orientation shifts of the environment map caused by axis conversions. While Blender and UE4 both use the z-axis as upward axis, only Maya uses the y-axis, as shown in Table 3. So, the orientation of the environment map was adapted to fit the z-axis as upward axis.

After creating the basic scene setup in Maya, the scene objects including camera, light data, and the sphere mesh, were transferred into the other rendering tools through an FBX file. Nevertheless, not all information could be fully restored in each software. Especially the implementation of the environment map differed in each tool. For example, emulating an environment map in Blender required to set an HDRi as a background inside the world settings, while the Arnold renderer in Maya has a dedicated scene object for placing environment maps, which is called *AiSkyDome*.

The most challenging implementation was the realization of environment maps inside the Unreal Engine. While the used software version also provides a prefabricated scene object for placing a sky dome, which is named *HDRI Backdrop*, this object behaves significantly different than environment maps inside the other rendering tools. First, the rotation of the environment map did not match the rotation of the other rendering tools. As already mentioned, the orientation

was adapted to fit the z-axis as upward axis. Nevertheless, a different rotation angle had to be entered in UE4 to approximate the framing of the other rendering tools. Unfortunately, a search for the cause of the changed rotation only showed that the angle does not correspond to the rotation angle of the other tools, neither in degrees nor in arcs. Consequently, the environment map in UE4 is slightly rotated differently and thus the rendered images of Unreal should be examined with caution since they do not display the same lighting situation as the other renderers. Another complicating factor is the not easy to handle brightness of the environment map in the Unreal Engine. There are two parameters influencing the intensity of the environment map, these are *Intensity* and the so-called *Lighting Distance Factor*. Through several test renderings I have found out that an *Intensity* of 0.5 and a *Lighting Distance Factor* of 1.0 best matches the scene appearance from other rendering tools. However, this matching was carried out without the calculations of the average mean brightness, as discussed in the previous section 3.3. So, it is likely that the scene brightness is not perfectly aligned with the other renderers. The last issue to mention is that the *HDRI Backdrop* requires a mesh for the projection of the environment map. Therefor a sphere mesh with inverted face normals and the same size as the visualized mesh for the *AiSkyDome* was modelled in Maya and imported into the Unreal Engine. It is unclear if the resolution of the used mesh has an impact to the scene lighting, but its position clearly influenced the resulting lighting situation. This is another factor that might make the rendered image different from the others.

3.5 Developing the sequential rendering process

The material chart, that shows the visual differences when varying a material parameter in one of the selected renderers, was compiled by several individually images. These images were generated in an automated process in which one material parameter at a time was increased over its entire value range with a specific step size before rendering the scene. The script for this process was implemented individually for each rendering software in the scripting language Python and can be found in the appendix A.1. It should be noted that for Blender and Maya the command line renderer without GUI was used. Afterwards, the set of images were put together in a row for each renderer and plotted against the other rendering tools for a qualitative analyzation of the visual disparities. The implementation for assembling the material charts was done in JupyterLab and is given in the appendix A.1 as well, including an interactive material chart plotter.

One difficulty in implementing the automated rendering process is discussed as it may have an impact on the resulting image. Unfortunately, it was not possible in the game engine UE4 to render the scene with a different camera than the editor camera with the chosen way of implementation. The taken approach was to start the application (i.e. game), increment the

chosen material parameter while in game mode, render the scene by taking a “high resolution screenshot” and automatically stop the application when the last value has been rendered. However, Python scripts can only start a level in the so-called “Simulate” mode, that is a game mode without any player controller. As the scene camera cannot be assigned to any player entity, only the editor camera is available. This is a disadvantage because the final image does not necessarily correspond to the result of a digital production. Nevertheless, the renderings should not differ too much since the editor option “Game View” was activated before starting the rendering process. This setting is meant to provide an in-game-like look for the editor. Eventually the implementation of this setting determines how close the result comes to a production-ready output.

4 Results

In analysing the rendered material charts, two methods were used. In an attempt to gain “prior knowledge of the semantic relationship among parameters” (p. 5) in the manner of Guarnera et al. (2018), the influences of each parameter were first compared to the ones of other parameters within the same renderer. This allows for conclusions about the intuitiveness of material design because the understanding of the effect produced should ideally be given by the parameter name itself. If the parameter effects different features than expected, the descriptor may be classified as unintuitive. Secondly, the impact of a parameter is compared across all rendering tools. This way it can be ascertained whether the material models in modern rendering pipelines evoke different images and if they vary in terms of quality and the range of reproducible materials. For example, certain renderers may not be able to produce a certain material feature at all.

4.1 Intuitiveness of parameters

4.1.1 Roughness

The parameter *Roughness* has a great impact to the appearance of a material. In fact, many characteristic features like specular highlights or reflections can only be observed if the surface is smooth. The influence of the parameter is exemplary shown in Figure 6 illustrating rendered images from Cycles in the lighting situation with one directional light. Lower values of *Roughness* correspond to smooth surfaces, while higher values correspond to a rougher surface. It is noticeable that the specular highlight of the depicted sphere first is increasing in area with reduced intensity and then vanishes with increasing *Roughness* approximately from the value 0.6. These observations coincide with the effective influence of the parameter on the brightness of the specular highlight depicted in Figure 7. The plot shows the intensity values of the green channel in the scan line 183 for each image shown in Figure 6. The intensity values of the green

channel are representative for the lightness of the specular highlight since humans are more sensitive to this colour channel, and in a simplified way it can be assumed that it corresponds to our perception of brightness. Each curve represents the brightness for a given *Roughness* value p_n, where the subscript character n corresponds to the given *Roughness* value. For example, $p_{0.1}$ corresponds to a *Roughness* value of 0.1. Both figures refer to the lighting situation with one directional light and no environment map, which is denoted as "environment: none". Also, it only illustrates the behaviour of *Roughness* in Cycles, but this plot is representative for the other render engines as well because in this lighting situation, all renderers returned similar results in my findings.

First, it should be noted that the intensity values of the highlight for the *Roughness* values 0.0 to 0.2 were clipped. This is because the reflected light of the specular highlight is outside the dynamic range. As Burley (2012) states, "highlights on shiny materials can reach into the hundreds" (p. 19) and require proper tone mapping or must be clipped, as Phillips et al. (2009) concur (p. 1). Yet, this topic will not be further discussed in the context of this paper. Secondly, it is noticeable that lower *Roughness* results in a higher-contrast image because the intensity values around the highlight at about 300 horizontal pixels change abruptly with a peak. Since higher contrasted images attract the eye more than low-contrast images, any change in this area will be immediately noticeable. This will likely have an influence on the intuitiveness of the parameter.

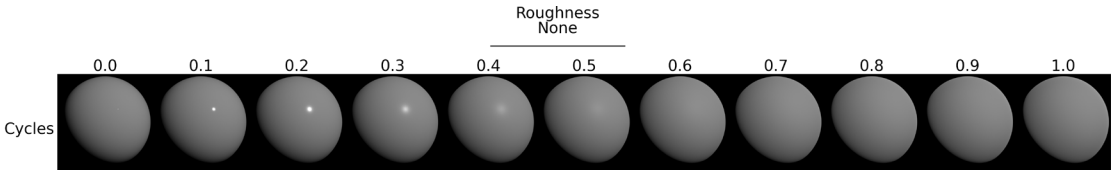

Figure 6: Varying "Roughness" in Cycles. The sphere is illuminated by one directional light instead of an environment map to clearly visualize the blurring impact on the highlight.

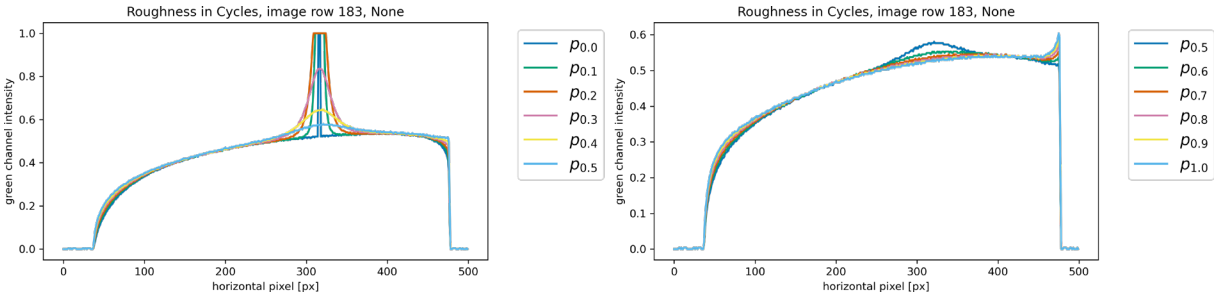

Figure 7: Influence of "Roughness" on specular highlights in lower and higher range values: "Roughness" from 0.0-0.5 on the left and 0.6-1.0 on the right. The plot shows the intensity values of the green channel in row 183 for the images shown in Figure 6.

Generally, surfaces are said to be rougher the more the highlight loses intensity and the wider it becomes in all rendering tools. This coincides with the laws of physics for specular and diffuse reflection. A smooth material surfaces tends to reflect the light only in one direction while diffuse surfaces scatter light in many directions, resulting in a wider and less bright highlight. So, the general impact of the parameter Roughness behaves very intuitively.

However, I state that the modification of *Roughness* in the lower range of 0.0 to 0.5 has a greater impact on material appearance than the values in the range of 0.5 to 1.0 in all rendering tools, because there is a greater change in brightness. This observation is also true for images rendered with the use of an environment map (see appendix A.1). Though, a perceptually linear effect of parameters is important for intuitive material design, as Burley (2012) and Karis (2013) assert. Even though no user studies have been carried out as part of this work, I claim that the parametrization of *Roughness* is not linear among the examined renderers based on the observations described above for this parameter. Nevertheless, this must be seen with cation as this statement is based on extremely limited data. The most important drawback is a lack of the exploration of the interrelations of parameters, as only one parameter is increased at a time. Hence, the observed non-linear behaviour of Roughness could result from a fixed value at 0.2 for *Specular*, for example. Still, a parameter should behave linear regardless of the values of other parameters. Therefore, I call the effect of the parameter *Roughness* not completely intuitive.

4.1.2 Metallic

The parameter *Metallic* influences whether a material is treated as conductor (i.e. metal) or dielectric (i.e. non-metal). This has a particular impact to the calculation of the Fresnel term and thus the specular reflectance of a material, as already mentioned in section 2.2. Because conductors generally have a higher reflectance than dielectrics, a higher value of *Metallic* causes a more reflective material as shown in the rendering results from Cycles in Figure 8. The final appearance of a metallic material thus depends more on its environment than a dielectric material would. The image for Metallic at 1.0 in Figure 8 depicts “simply a distorted reflection of the world around it” (p. 347), as Fleming et al. (2003) described this effect.

In terms of the intuitiveness of material design, the parameter *Metallic* causes expected results as we are used to metallic surfaces being more reflective. Though, in the example shown in Figure 8, the *BaseColor* of the material is no longer visible at first sight, which causes the question, how intuitive colour design for metallic materials is. Because the effects of the *BaseColor* have not been examined in detail, the following statements should be interpreted with caution. First, it is important to know that while dielectric materials use *BaseColor* to calculate the diffuse albedo that is perceived as the colour of the object, smooth metallic surfaces have no diffuse reflection component. Rather, the information for the colour of the metallic surface cannot directly be inferred from the appearance of the object as it is intertwined with the reflected image content of

the scene. For example, in Figure 8 the brown tones occur naturally because of the wooden floor of the living room in the environment map "Lebombo". However, if the *BaseColor* of the portrayed sphere was not purely white as in all my findings but instead red, for example, it would have been noticeable that the sphere appears reddish. It can be assumed that the *BaseColor* of a metallic surface is still perceivable at least for colours different from white or black because although smooth metallic surfaces do not have a diffuse albedo, they have "spectral (mean use RGB color) specular reflectance" (Lagarde, 2011a), meaning that certain wavelengths of light are more reflected than others, which results in the perception of a reflected colour (Hecht, 2018, p. 268). In contrast, dielectric materials "have white specular reflectance" (Lagarde, 2011a), which means that the incident light is not changed in wavelength, so the colour of the incident light is not changed by the material. Nevertheless, in the real world most metals including aluminium, tin and steel appear achromatic like the sphere in Figure 8 because the incident light is reflected regardless of the wavelength in equal parts (Hecht, 2018, p. 266). This corresponds to the behaviour of dielectric materials except for the missing diffuse reflection for smooth surfaces. Hence, the colour of the object is not attributed to the material because it only reflects the colours of its surroundings. In conclusion, colour design for metallic surfaces in a PBR workflow is generally intuitive if the spectral behaviour of specular reflections is considered in the implementation of the material model. The parameter *BaseColor* still affects the colour representation of the material although it changes other material properties than it does with non-metallic materials.

Eventually, it may not be clear to the user that intermediate values between 0.0 and 1.0 are only intended to enable interpolation of materials. For real-world materials, there is no intermediate state between metals and non-metals. The interpolation between those two states allows for a mixture of both material reflection properties but may also lead to errors when reproducing realistic materials. For designers, it is important to understand that *Metallic* is not primary a parameter to make the object look glossier or more reflective. Especially the latter property should preferably be changed by varying *Roughness* or *Specular* instead. What *Metallic* does is to decide whether a smooth material surface generally has a diffuse reflection component or not. If the material does not reflect diffusely at all (i.e. metallic), the material certainly appears more specular as well, but it is a fundamental change in the material properties. This subtle difference in processing the material properties requires a trained eye of the designer and remains quite complex. Therefore, the parameter metallic is not intuitive on closer examination, because the intermediate values for *Metallic* may lead to a wrong understanding of its influence.

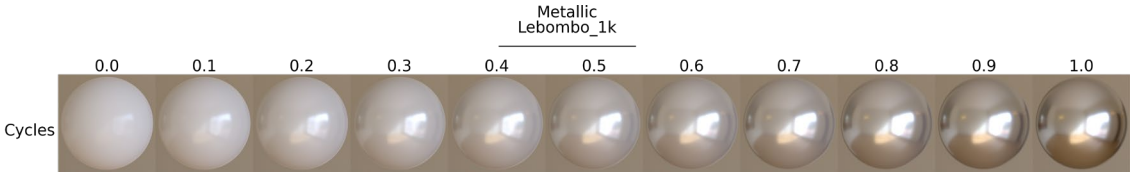

Figure 8: Varying "Metallic" in Cycles in environment "Lebombo"

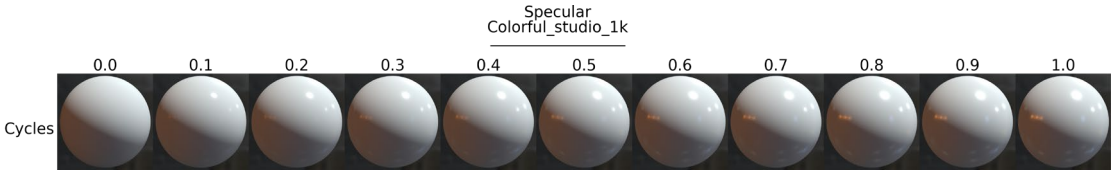

Figure 9: Varying "Specular" in Cycles in environment "Colorful Studio"

4.1.3 Specular

The *Specular* value appears to influence the intensity of all specular reflections. It can easily be mistaken for *Roughness* at first glance, however the parameter *Specular* does not affect the fuzziness of the specular highlight. Instead, it seems to act like a multiplier for the intensity of the highlight.

Although this chapter does not deal with the comparison across rendering tools yet in detail, it is important to note that the *Specular* parameter in Arnold has a different meaning than in the remaining tools. According to the reference implementation for the "Autodesk Standard Surface" shader provided by Georgiev et al. (2019b), the *Specular* parameter indeed acts as a multiplier for the intensity of the specular reflection (l. 126-129). As already mentioned in section 3.2.4, the IOR for calculating the Fresnel term is exposed as a separate parameter in the Arnold renderer. In contrast, the IOR of the material in Cycles, Eevee and Blender is controlled indirectly by the *Specular* parameter. A conversion between *Specular*, F_0 and the IOR is provided by Burley (2015), based on the formulation for F_0 by Cook and Torrance (1982, p. 16), and adopted by the mentioned renderers (Blender Foundation, 2020b, l. 101; 'ShadingCommon.ush' in Epic Games, 2020c, ll. 76–97). More details on this will be given in the appendix A.2.4.

In terms of intuitiveness, Burley (2015) endorses that the IOR as a separate parameter "was considered unintuitive for artists" (p. 13). This is probably because the name comes from the field of physics and is not intuitive for people unfamiliar with this terminology. Hence, the material design in the renderers Cycles, Eevee and Unreal is more intuitive, as artists do not have to deal with setting the IOR for a material in terms of reflection. However, one could argue that the *Specular* parameter in Arnold allows for a more traditional control over the amount of specular reflections. Like a layer in Photoshop, the specular highlight can be precisely blend in and out. While this might be less physically plausible, it allows for greater and easier artistic control, as this kind of interaction is commonly used in design software. Eventually, the greatest issue in

terms of intuitiveness is that the *Specular* parameter share the same descriptor but affect material features in a fundamental different manner. This can easily get confusing for the user as the semantics of this parameter is non-identical across the examined tools. In section 4.2.3 it is discussed how extensive the visual disparities are in practice.

Finally, all renderers have in common that the *Specular* parameter only has any impact if the material is non-metallic (i.e. dielectric). For example, the Autodesk Standard Surface shader, which "follows the design of the Standard Surface shader in the Arnold renderer" (Georgiev et al., 2019a), distinguishes between a "metal_brdf [and a] specular_reflection_layer" (Georgiev et al., 2019a, Chapter 3.4). Thus, metallic materials are only influenced by the *BaseColor* and the special parameter *SpecularColor* in Arnold. Both influence the spectral specular reflectance, as described in section 4.1.2 and following the findings of Gulbrandsen (2014). Nevertheless, the *Specular* parameter has no impact on metals in Arnold. In the semantically differently used parameter in Cycles, Eevee and Unreal, *Specular* does not have any effects on pure metallic materials, either. Instead, both different material models for dielectric and metallic materials are linearly interpolated. This has already been discussed in section 3.3 with a detailed view to the UE4 source code as an example. However, I maintain that this parameter concept is not the most intuitive because the term "specular" alone is not self-explanatory for the fact that this parameter only affects non-metallic materials. In fact, metallic surfaces only have a specular reflection component as explained before, so it is misleading that the *Specular* parameter has absolutely no effect on metals.

4.1.4 Clearcoat

The impact of *Clearcoat* is very subtle in the examined lighting situations. As the effect is barely noticeable in small figures, no material chart is presented at this point. Please refer to the appendix A.1 for all images in full resolution. The impact of *Clearcoat* is best observable in my findings with the renderer Arnold viewed in the lighting situation "Lebombo" or "Colorful Studio".

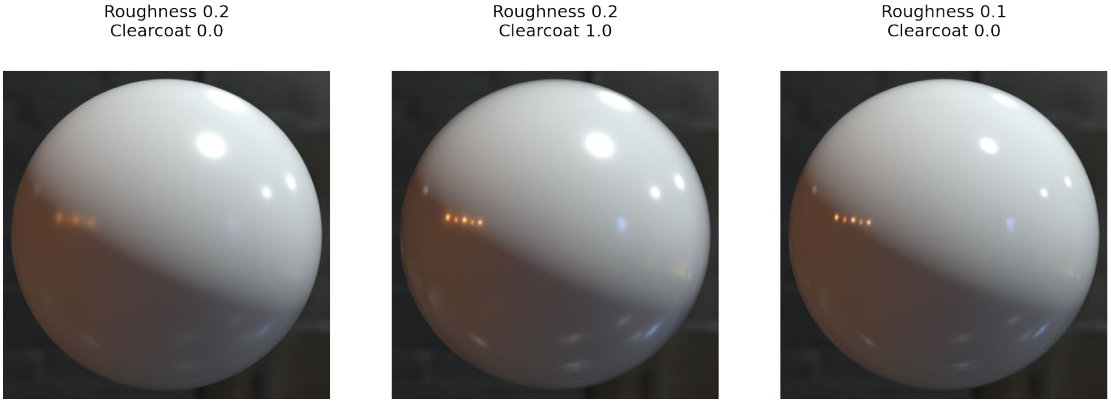

Figure 10: "Clearcoat" vs. "Roughness" in Arnold (lighting situation: "Colorful Studio")

The effect of *Clearcoat* can easily be mistaken for *Roughness*, as both influence the sharpness of the specular highlight. However, *Clearcoat* acts as a second layer on top of the base material. Thus, the underlying blurry highlight typical for rough surfaces is still perceivable, although there is a specular layer on top of it. Figure 10 depicts this effect and opposes the effect of *Roughness* and *Clearcoat* in Arnold in the lighting situation "Colorful Studio". The middle image is the only one with the *Clearcoat* effect on it and appears as a blend of the left and right image. Especially the area around the orange specular reflection looks very blurry in the left image but gets sharper when adding a clearcoat layer in the middle image. Nevertheless, in comparison to the right image without a clearcoat layer, the orange specular reflection is still more blurry in the middle image. This is because the coating acts as another smoothening surface that is "always reflective (with the given roughness) and is assumed to be dielectric" (*Coat - Arnold for Maya User Guide - Arnold Renderer*, n.d.), at least in the Arnold renderer.

The intuitiveness of the parameter *Clearcoat* can only be assessed based on limited data. In my results, the effect is barely noticeable in the given lighting situations. Also, the results need to be interpreted with caution as additional properties of the clearcoat layer like its roughness or IOR have been disregarded in the context of the investigations. Based on these observations I would describe the parameter as not fully intuitive since the effect is so subtle that it is hard to use the effect purposefully. Nevertheless, its descriptor is clear in its meaning and thus it is possible to define a material without seeing the effect directly in the viewport window (at least to some degree, if the parameter set is considered as an abstraction of the material properties).

4.2 Comparison of parameter effects between different renderers

4.2.1 Roughness

When varying the parameter *Roughness*, the results are quite different across each renderer. For a clean impression of the parameter impact, the images were first compared in the lighting situation without any environment map and a directional light only. While at the first glance the images appeared similar, certain differences were noticeable especially when looking at the alteration at every step.

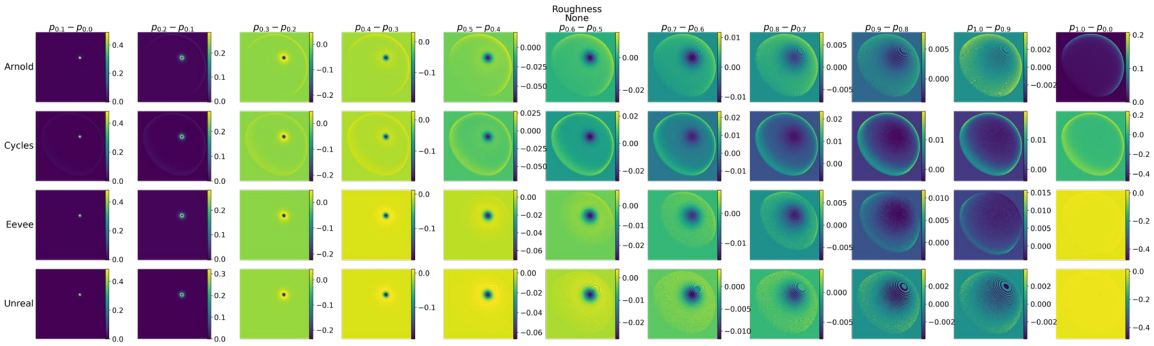

Figure 11: Cross-renderer material difference chart for "Roughness" without environment map. The last image shows the difference image between value 0.0 and 1.0

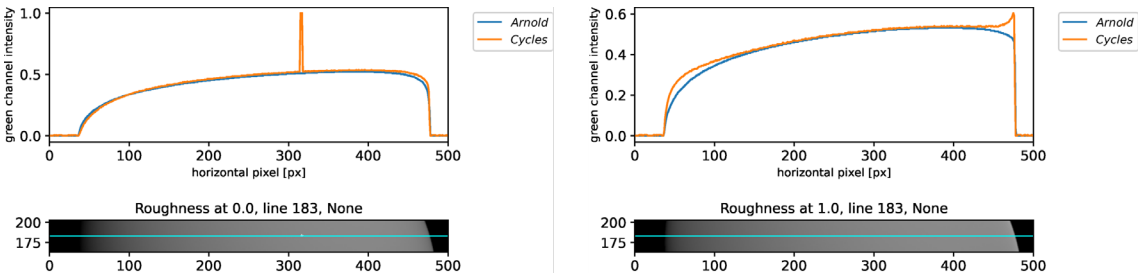

Figure 12: Brightness plot for "Roughness" 0.0 (left) vs. 1.0 (right) in Arnold and Cycles. The image below shows the plotted image line 183 (cyan) in the context of the total image, rendered in Cycles.

Figure 11 depicts a material chart that shows the difference images between each incremental step of changes in *Roughness*, starting with the difference between a *Roughness* value of 0.0 and 0.1 and ending with the difference between 1.0 and 0.9 in the second last column. The last column displays the difference between the highest and lowest *Roughness* value, that is the difference between 1.0 and 0.0. As the background image remains the same in each row, the difference of this area is always zero. Hence, the colour of the background provides a reference for evaluating the changes in brightness in the image. Brighter areas correspond to an increase in brightness in this area, while darker areas, compared to the background, visualize a drop in image brightness caused by the change of the parameter.

In the evaluation of the difference images, the most striking disparity among the renderers is the handling of light at grazing angles and around the terminator. While both path tracers Arnold and Cycles show an increase in brightness towards grazing angles in Figure 11, this appearance feature is not present in the two rasterizers. In a detailed look at the differences between Arnold and Cycles, it is also noticeable that Cycles considers retro-reflection, that is a higher reflectance at grazing angles than at normal incidence, whereas Arnold does not reproduce this effect. Instead, the maximum reflectance at grazing angles in Arnold seems to be limited to the diffuse colour at normal incidence (i.e. F_0). This behaviour is also observable by plotting the intensities of the rendered images. Figure 7 already illustrated the changes in brightness in line 183 of the

image for all *Roughness* values in the current examined lighting situation. Figure 12 extends this presentation by contrasting the two renderers Arnold and Cycles for the *Roughness* values 0.0 and 1.0, which corresponds to an ideal smooth and purely rough surface. For an ideal rough surface, it is noticeable that Cycles returns higher brightness values at around 475 horizontal pixels, which corresponds to the edge of the sphere. This peak in brightness at grazing angles is even higher than the general diffuse colour of the sphere and might be interpreted as retro-reflection. As Burley (2012) has noted, "rough surfaces tend to have a peak instead of a shadow" (p. 6) towards grazing angles if they are non-metallic (otherwise they would not have a diffuse component). Interestingly, only Cycles appears to model this phenomenon correctly in the lighting situation with one directional light.

However, this observation is not true for the lighting situation "Lebombo", as depicted in Figure 13. What is even more interesting is that Eevee and Unreal seem to consider the effect of retro-reflection, but exactly the other way round. According to Burley (2012) this is not physically plausible. Though, a detailed examination of this issue is beyond the scope of this paper.

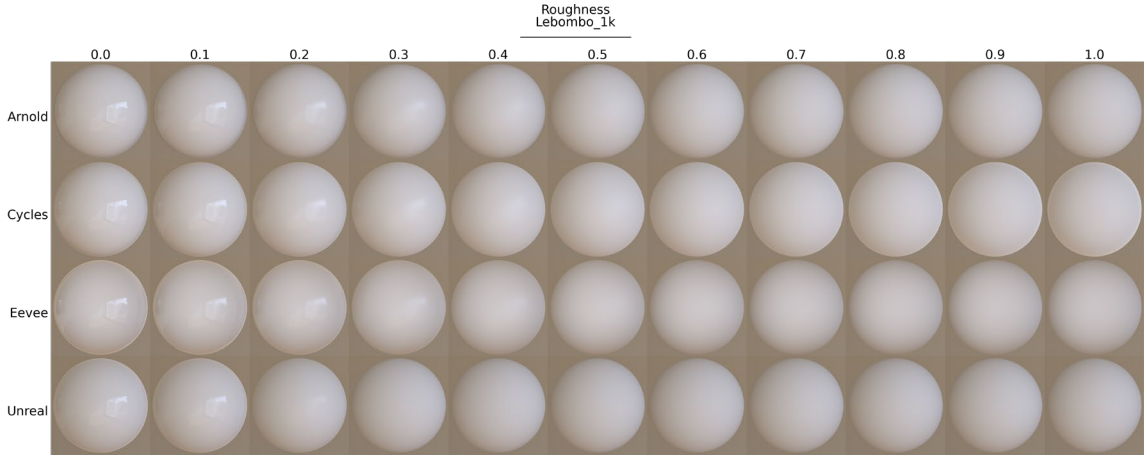

Figure 13: Cross-renderer material chart for "Roughness" in environment "Lebombo". Notice the difference behaviour at grazing angles: Cycles becomes lighter with increasing roughness, while the other three renderers get darker towards grazing angles.

4.2.2 Metallic

The most noticeable difference for the parameter *Metallic* across all renderers was the significantly darker appearance of metallic materials in UE4. Figure 14 shows the material chart for varying *Metallic* from 0.0 to 1.0 across the different renderer in the lighting scenario "Lebombo". The perceived colour of the sphere with a metallic surface in the last column is remarkably darker in the results from Unreal compared with Arnold, Cycles and Eevee. To exclude the possibility that the phenomenon is caused by different light intensities of the environment map, Figure 15 shows the intensity of the green channel in line 190. While Unreal has slightly lower intensity values in this lighting scenario than the other renderers, all rendering

tools approximately start with the same intensity values. The small deviation occurs probably because the light intensities in environment maps have not been matched like for the scene with one directional light source. Nevertheless, there is a significant decrease in the intensity values for metallic materials in UE4 compared with the other renderers. The difference in intensity of the metallic material surface between Arnold, Eeevee and Cycles, that produces really similar results in this lighting situation, and Unreal is about 0.1 in this environment. The other lighting settings "sunflowers" and "moonless_golf" produce similar results, only the environment map "colorful_studio" has evoked similar intensity values across all renderers. The material charts and intensity plots for these settings are given in the appendix A.1. Unfortunately, I could not find a sufficient explanation for the deviating test results of Unreal in the mentioned light situations.

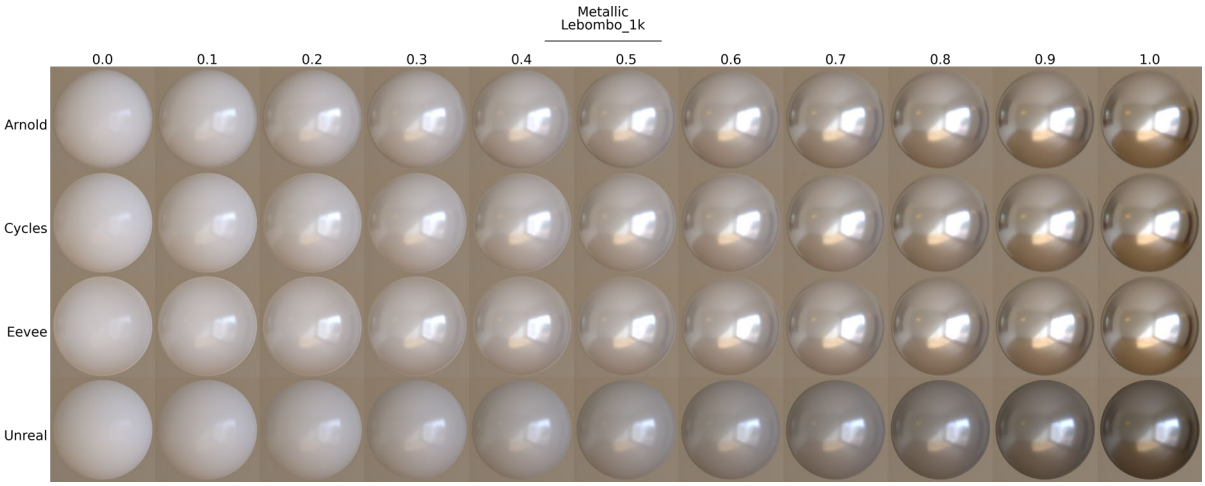

Figure 14: Cross-renderer material chart for "Metallic" in environment "Lebombo"

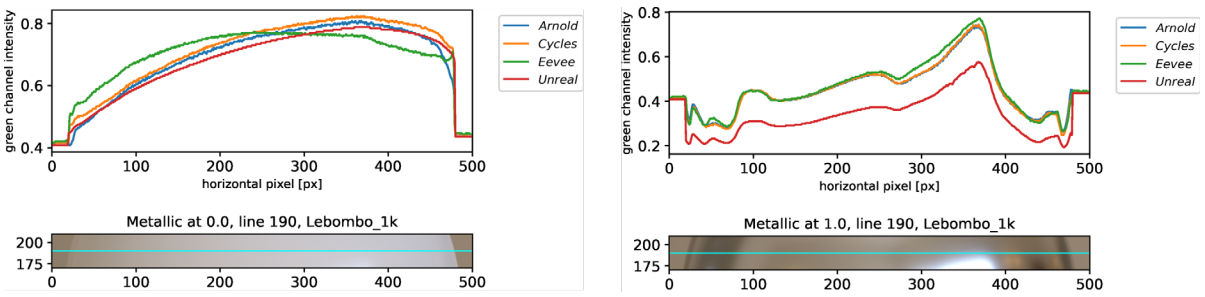

Figure 15: Intensities for dielectric vs. metallic surface across all renderers with noticable drop-off in UE4. The image below shows the plotted image line 190 (cyan) in the context of the total image, rendered in Cycles.

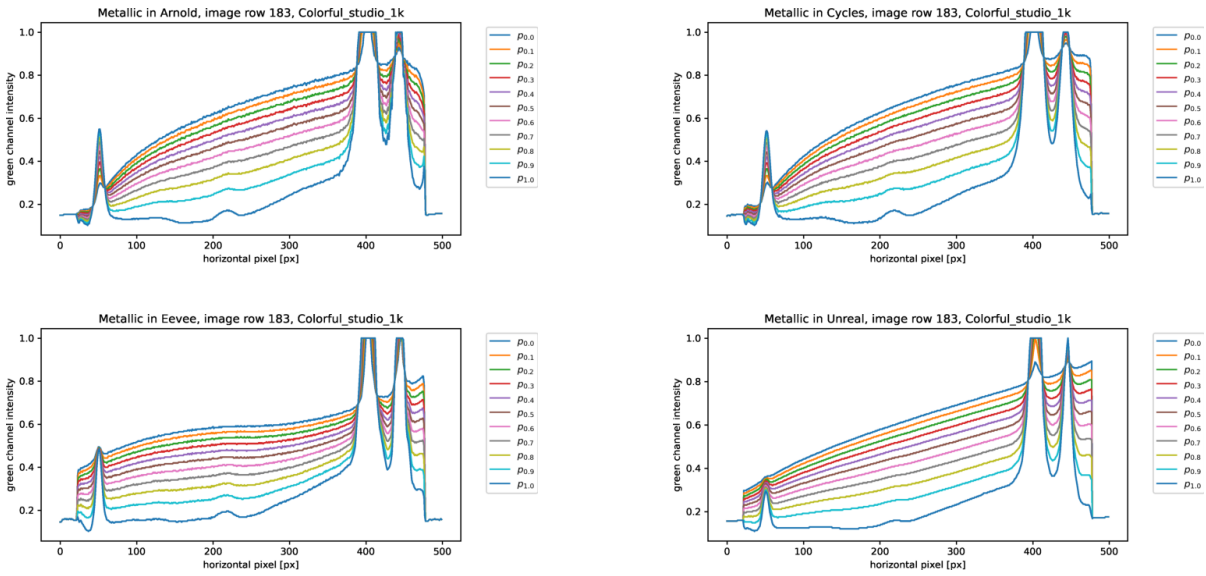

Figure 16: Linear parametrization of "Metallic" across all renderers. The plot shows the intensity values of the green channel in row 183 for the images taken in environment "Colorful Studio"

Nevertheless, the parameter *Metallic* is linear parametrized in all renderers. This statement was made based on Figure 16, as it shows an evenly variation in brightness for all renderers when increasing the value of the parameter. The linear parametrization might be due to the fact that intermediate values for Metallic are usually interpolated linearly, as described in section 3.3 for UE4. This is of advantage for material design because a steady change of the parameter corresponds to an even visual change in material appearance what generally meets the expectations of the user (cf. Burley, 2012; Karis, 2013).

4.2.3 Specular

As already explained in section 4.1.3, the parameter *Specular* in Arnold changes a fundamentally different appearance feature compared to the other three rendering tools. This is also conspicuous in a detailed examination of the rendering results. Figure 17 shows the change in brightness when varying the parameter *Specular* in all renderers in the environment "Moonless_Golf" for the scanline 255. First, it is noticeable that the intensity of the green channel in Arnold increases linearly in all areas. This is likely because the specular parameter is used as a multiplier in the calculation of the specular part of the reflection, as explained in section 4.1.3. In contrast, the image intensity of the other renderers does increase non-linearly at grazing angles. There is a noticeable jump when changing *Specular* from 0.0 to 0.1 around the edges of the sphere at about 50 and 450 horizontal pixels.

The sudden rise at grazing angles is also visible when looking at the difference images between each variation step of *Specular*. It is best observable in the lighting situation "Lebombo" and depicted in Figure 18. At the inner areas of the sphere, however, the increase in brightness appears linear as well, like shown in Figure 17 in the range between 100 and 400 horizontal

pixels. Also, Figure 18 shows the same unchanged difference image for each variation step of *Specular*, which means that the change between each parameter tweaking is the same. So, all renderers behave linearly in the inner area of the sphere but differ in the representation towards grazing angles when varying the parameter *Specular.*

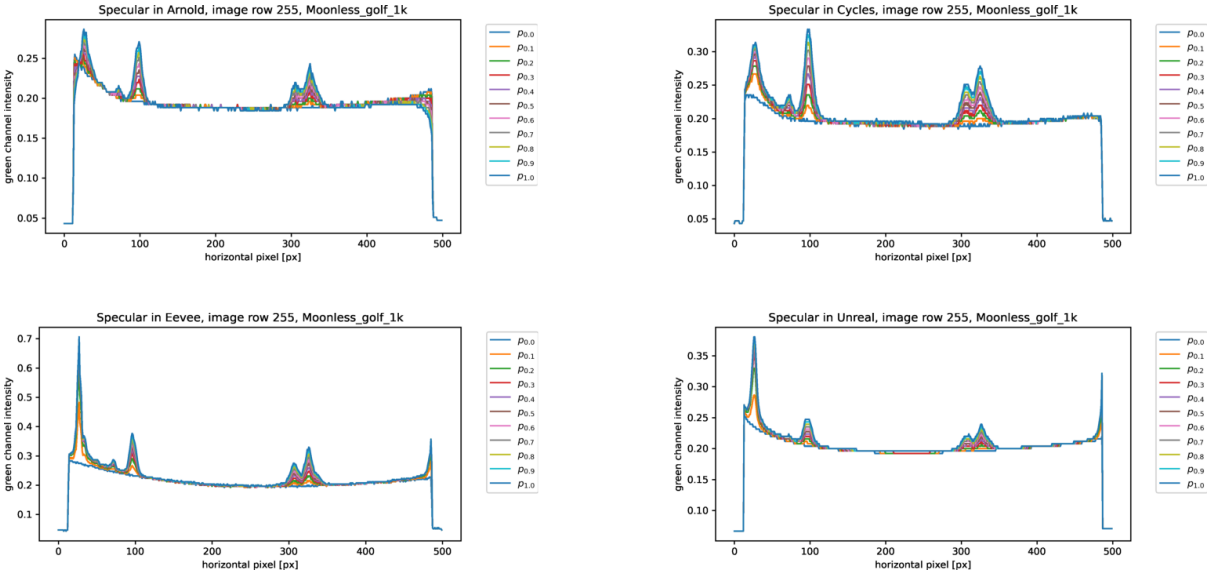

Figure 17: Brightness plot for "Specular" in environment "Moonless Golf" across all renderers. While Arnold shows a linear increase in brightness, the other three renderers behave non-linear at grazing angles.

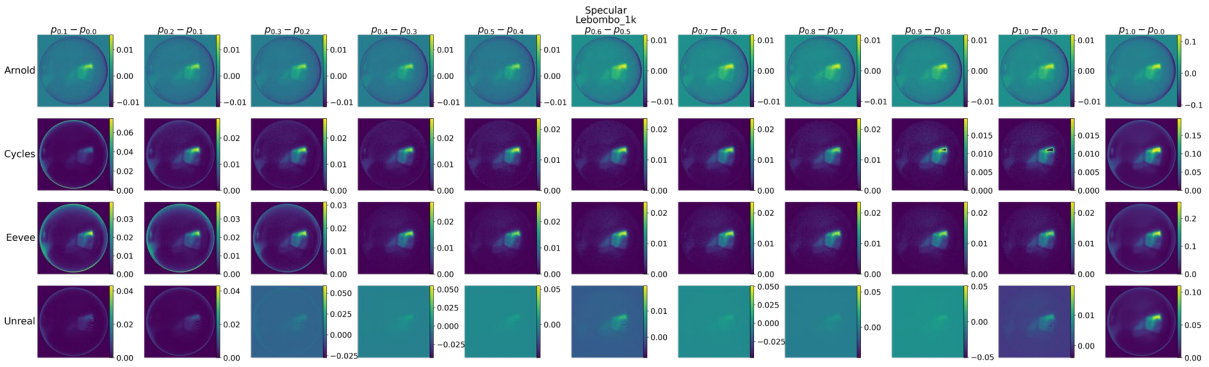

Figure 18: Cross-renderer material difference chart for "Specular" in environment "Lebombo" . The last image shows the difference image between value 0.0 and 1.0

It is interesting that Arnold continuously gets darker towards grazing angles with an increasing value for *Specular*, while the remaining rendering tools tend to get lighter, that is more reflective, towards grazing angles. It can be assumed that this is due to different calculations of the Fresnel term, because according to Burley (2012) "the grazing shadow for smooth surfaces is predicted by the Fresnel equations" (p. 6). As *Specular*, however, only changed the intensity of the specular reflection, the *Roughness* of the surface remains the same, which is at 0.2 for a nearly smooth surface. Hence, the outer line of the sphere should have a grazing shadow according to Burley (2012). My findings in section 4.2.1 already showed that only Cycles models this behaviour

correctly. But what it interesting now is that the shadow at grazing angles in Arnold is part of the specular reflection. A higher value for *Specular* intensifies the wrongly modelled grazing shadow. In contrast, a *Specular* value of 0.0 cancels the effect of specular reflection and hence the Fresnel effect completely in the renderers Cycles, Eevee and Unreal. As soon as the *Specular* value is increased, the reflectance at normal incident angle F_0 is greater zero and the Fresnel effect causes a brightening towards grazing angles. This might be a possible explanation for the sudden rise at grazing angles for the renderers Cycles, Eevee and Unreal.

In sum, the visual disparities among the renderers for the parameter *Specular* are minimal in my findings and mostly noticeable towards grazing angles. This is likely due to a different implementation of the Fresnel term among all renderers and is correlated with the effect of *Roughness*.

4.2.4 Clearcoat

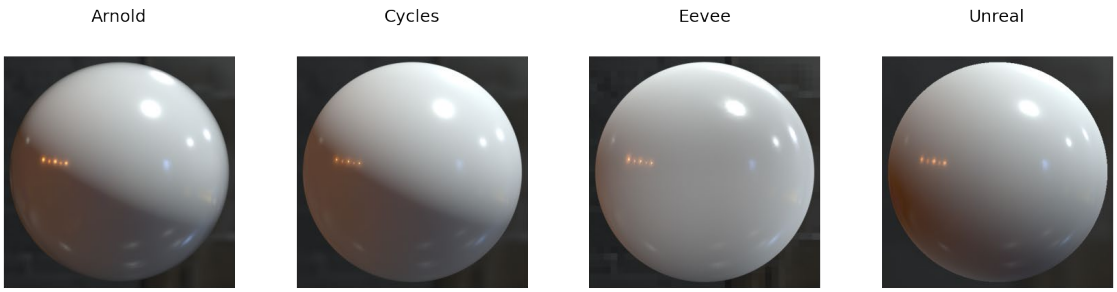

Figure 19: "Clearcoat" at 1.0 across all renderers in environment "Colorful Studio"

The parameter *Clearcoat* evokes vastly different results across all renderers. The reason for this is probably that this material feature allows a certain margin of interpretation. For example, Burley (2012) describes the feature as “a second, special-purpose specular lobe” (p. 13). He later specifies it to be “always isotropic and non-metallic” (p. 15). In contrast, the Arnold renderer allows for an anisotropic clearcoat layer, for instance (*Coat - Arnold for Maya User Guide - Arnold Renderer*, n.d.). Additionally, it is a rather complex material appearance feature to describe.

For this parameter, the examination of the scanline plots as seen in the previous sections was rather unrevealing. The overall change in brightness on the image was barely noticeable in these plots except minor variations around specular reflections of the sphere. However, the difference image material charts revealed a deeper insight of the behaviour of this parameter. Figure 20 displays the difference image when varying *Clearcoat* in 0.1 steps in the lighting situation “Lebombo” for all renderers. The illustration suggests that the parameter is evenly parametrized among all renderers since the difference image does not drastically change in any row of a renderer. Secondly, the overall impact of the parameter *Clearcoat* is greatest in Arnold and

Unreal, while the results of Cycles and Eevee show almost no change in brightness. Moreover, the impact of the parameter again adds a grazing shadow in the Arnold renderer while the results of the remaining rendering tools show an increased reflectivity towards grazing angles with a greater value for *Clearcoat*. In Cycles and Eevee there is even no decrease in brightness at all, as the lowest value of the colour bar in Figure 20 for these renderers is permanently zero. This again suggests that the Fresnel term is implemented in different ways to model the clearcoat layer in the different material models. And due to the linearized parametrization, it can be assumed that the calculated Fresnel effect is faded in by linear interpolation, just like the impact of the parameter *Metallic*. Nevertheless, these are just assumptions based on little data and they should be understood with care.

Taken all together, only a few statements can be made about the parameter *Clearcoat*. In some lighting situations the impact is more visible than in others and in some, the caused change is not perceivable at all. More importantly, the implementation of this material feature varies widely among the examined rendering tools. The most appealing and physically-plausible look was achieved with the Arnold renderer, as this was the only tool that considered the grazing shadow observed by Burley (2012) for smooth surfaces. However, the result may have been distorted by disregarding the roughness and IOR of this second specular layer.

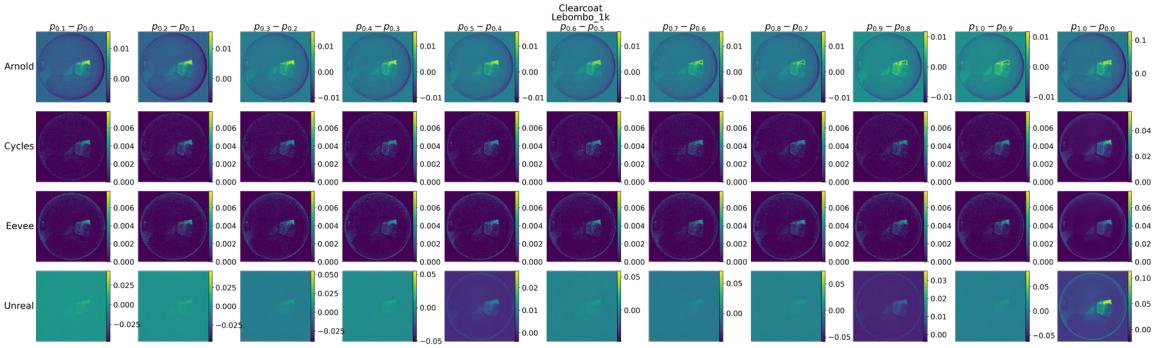

Figure 20: Cross-renderer material difference chart for "Clearcoat" in environment "Lebombo"

5 Limitations

This research has several limitations that highlight the wide-ranging extent of the subject. First, there are many other commonly used rendering tools that could not be examined in the scope of this paper. To mention a few, these include the game engine Unity, the material and texture editing software Substance Painter or the procedural generation software Houdini. Also, the path tracing feature of the Unreal Engine has not been surveyed in this paper. It remains for future work to analyse the differences in material models between these software. Moreover, algorithms for material models are in a continuous change since the improvement of material models is an active field of research. So, the findings of this work might be outdated in a few years already.

To gain a more comprehensive picture of the intuitiveness of material design, more than four parameters need to be studied. The influences of material parameters in the current used interfaces are not always evident from their description. Often, they are mutually dependent and further data collection is required to determine exactly how one parameter affects another. This includes the examination of the perception of colour reflected by a material surface and whether a special colour can emphasize certain material appearance features in a better way. For example, the question arises of whether a coloured material chart like Burley's (2012) would have shown the differences between the renderers more clearly.

Additionally, only four different lighting scenarios are considered. The examinations lack the acquisition of low contrast light situations since the environment maps were chosen at random rather than well justified. This is because I had not found enough time to deal with the topic of the influence of light on appearance design. Hence, by far not all material features have been sufficiently studied in this paper. Moreover, because only reflections were analysed, features like anisotropy, transmission or absorption are completely missing from the considerations in this paper. This applies to the study of the range of reproducible materials as well. The formulation of a gamut for materials would have exceeded the scope of this work, as it is a not trivial, high-dimensional problem since the BRDF alone has four dimensions to consider. However, it is an exciting topic for future research.

Furthermore, no brightness adjustments for environment maps were undertaken, as already mentioned in section 3.3. The influence of the resolution of the environment maps has not been investigated yet, too. Both factors may cause misleading brightness values and thus the interpretation of the presented results should be understood with caution.

Another aspect to consider in future work may be the use of different object shapes for observing different material appearance features. In a recent study, Schmidt et al. (2020) state that "the shape of objects and surfaces is an important cue for material perception and operates at different spatial scales: microscale (surface roughness), mesoscale (textures or local object shape), or megascale (global object shape)" (p. 15). In this paper, the influence of meso- or megascale shape has been neglected. But especially in polygonal virtual worlds, the influence of shapes certainly plays a decisive role, not least because each renderer uses a slightly different light transport algorithm.

Finally, the intuitiveness of material design is also highly dependent on the graphical user interface. While Schmidt et al. (2014) highlighted several more intuitive interfaces for material editing, the current rendering software still rely on the conventional direct interface for the design process. In the context of this paper the interfaces were not extensively investigated, but it is certainly exciting to find out whether the interrelations of light, shape and material parameters can be more clearly represented by an improved user interface concept.

6 Discussion

My findings showed that modern rendering software has mainly adopted the same PBR workflow. The main differences occurred due to dissimilar methods for calculating light transport itself. For example, the renderers Arnold and Cycles rely on path tracing and generally produce more realistic results as the two rasterizers Eevee and Unreal. This was especially noticeable in the missing terminator in Eevee in the lighting scenarios "Colorful Studio" and "Lebombo", but also in the lower visual quality of specular reflections in Eevee and Unreal, compared to Arnold and Cycles. Nevertheless, there are some minor disparities because of non-identical underlying material models, resulting in dissimilar intuitiveness of material editing and final scene appearance in each tool.

The parameter *Specular* differs in its syntactical meaning between Arnold and the other tools. Arnold treats the parameter as a multiplier to all specular reflections and provides a separate parameter to set the IOR. Additionally, it is possible for the user to define a specular colour regardless of the conductivity of a material. This allows for non-physically plausible dielectric materials that change the colour of incident light. In the real-world, only metals can change the wavelength of light when reflecting it. Based on this assumption, it is not possible to specify a specular colour in the rendering tools Cycles, Eevee and Unreal. Moreover, the *Specular* parameter influences the IOR, meaning the reflectance at the normal incident angle at 0°, rather than the overall brightness of specular reflections. While the latter concept is more robust in terms of physically credibility, it also restricts creative freedom of design. The group of non-metallic materials with a tinted highlight cannot be represented in the material models of Cycles, Eevee and Unreal, as they are classified as physically implausible. Also, the *Specular* parameter only has an impact on non-metallic materials in all rendering tools. This may be confusing since this coherence is not communicated in the UI well enough in the examined tools.

The calculation of metallic materials is remarkably different in Unreal. In my findings, the resulting material surfaces were significantly darker in comparison to other renderers. Unfortunately, I was not able to find out the reason for this yet.

Roughness of materials is interpreted in slightly different ways across all renderers. This is likely due to distinct implementations of the Fresnel effect. While Arnold seems to consequently add a shadow towards grazing angles for smooth surfaces, Cycles fails at considering this effect when adding a smooth *Clearcoat* layer. However, Cycles appears to be the only tool modelling retro-reflection and grazing angles correctly in the sense of Burley (2012). Interestingly, both real-time renderers Eevee and Unreal seem to model the Fresnel effect in a simplified way that misses the grazing shadow for smooth surfaces. Instead, the edge of the sphere even appeared brighter

towards grazing angles which reminds of the phenomenon of retroreflection. However, this effect is exactly inverted to its real-world behaviour.

Eventually, all listed disparities are subtle and mostly not perceptible when working within one rendering tool. However, these circumstances compound a seamless exchange of data between rendering tools among different departments in digital productions. The output renderings vary in the reproduction of special material features or might not be able to reproduce a material in another renderer at all, such as dielectric materials with tinted highlights.

Additionally, the question of whether there is a correct or incorrect way of defining materials in a PBR workflow is left open. There certainly are criteria by which one can judge how realistically a real-world material is reproduced in a model. But especially in digital media productions, it is often not the goal to create a world that already exists. This leads to the question of how much creative freedom is left to the artist in a PBR workflow. Burley (2012) successfully demonstrated the integration of the PBR workflow into their rendering pipeline while preserving the characteristic art style of Disney. He showed that PBR workflows not necessarily imply photorealism. The question is, when it makes sense to follow physical principles, when the workflow is hampered and when PBR even sets limits to creativity in the worst case. Stum et al. (2016) claim that "one cannot use arbitrary inputs for the reflectance values. […] The values must be correct and measured from real world data" (p. 120). While I agree that the values should be set with a certain intention, I argue that parameter values should not be constrained to measured values, at least in the context of digital media productions. Instead, I suggest integrating keywords that describe the current material appearance in the UI to make the user aware of the effect of the parameters in different lighting situations. For example, the influence of the parameter IOR may not be visible immediately. An info box could indicate that the current parameter value corresponds to the IOR of water. Additionally, different lighting situations to view the effect of the parameter may be suggested in the UI.

7 Conclusion

The appearance of materials is a complex interplay of light, shape, and material properties. Modern rendering tools already allow to emulate the most significant material features, resulting in a photorealistic reproduction of reality. However, by far not all material features can be depicted efficiently yet. It is an active and recent field of research to find out how humans perceive materials and their appearance, how light is transported through the scene and interacts with surfaces and eventually, how to translate the gained knowledge into intuitive interfaces for material editing. It remains a challenge to intuitively convey the interplay of light and material features, especially since certain characteristics can only be observed in special lighting situations and from different viewing angles. An exciting idea to address this problem could be the application of VR as it enables to view the light transport on material surfaces in a more natural way. However, currently the use of VR also limits computational complexity and hence it takes additional effort for an efficient implementation of the shading model.

This paper showed that there is inconsistency among current rendering tools when it comes to capturing the material properties abstractly in a model. Every renderer has its own realisation, and it will likely remain a challenge to provide a seamless exchange between these pipelines for digital productions working with miscellaneous tools in different departments. Nevertheless, the results of the examined rendering software are quite similar with some exceptions. The most significant visual disparities still occur due to different rendering mechanisms.

In future works a standardized material model may be developed to simplify the process of exchanging material data among renderers. Additionally, a material model adapted to the individual strengths of a renderer could represent a highlighting feature for this software.

References

Adelson, E. H. (2001). On seeing stuff: The perception of materials by humans and machines. *Human Vision and Electronic Imaging VI*, *4299*, 1–12. https://doi.org/10.1117/12.429489

Autodesk. (2020). *Maya* (Version 2020) [Computer software]. Autodesk.

Blender Foundation. (2013, August 10). FBX binary file format specification. *Blender Developers Blog*. https://code.blender.org/2013/08/fbx-binary-file-format-specification/

Blender Foundation. (2017, July 16). *Principled BSDF — Blender Manual* [Images of material charts]. https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/principled.html

Blender Foundation. (2020a). *Blender* (2.83.1) [Computer software]. Blender Foundation.

Blender Foundation. (2020b). *Cycles—Node_principled_bsdf.osl* (2.83.1) [Source Code]. Blender Foundation. https://github.com/blender/blender/blob/master/intern/cycles/kernel/shaders/node_principled_bsdf.osl

Blender Foundation. (2020c). *Eevee—Bsdf_common_lib.glsl* (2.83.1) [Source Code]. Blender Foundation. https://github.com/sobotka/blender/blob/master/source/blender/draw/engines/eevee/shaders/bsdf_common_lib.glsl

Bousseau, A., Chapoulie, E., Ramamoorthi, R., & Agrawala, M. (2011). Optimizing Environment Maps for Material Depiction. *Computer Graphics Forum*, *30*(4), 1171–1180. https://doi.org/10.1111/j.1467-8659.2011.01975.x

Burley, B. (2012, August). *Physically Based Shading at Disney* [Conference session updated version 3]. SIGGRAPH 2012 Course: Practical Physically Based Shading in Film and Game Production, Los Angeles, California. http://disney-animation.s3.amazonaws.com/uploads/production/publication_asset/48/asset/s2012_pbs_disney_brdf_notes_v3.pdf

Burley, B. (2015, August 12). *Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering* [Course notes]. SIGGRAPH 2015 Course: Physically Based Shading in Theory

and Practice, Los Angeles, California. https://pdfs.semanticscholar.org/569b/15db4b472d430293728d44dfe81c05fac888.pdf?_ga=2.194743342.2029658154.1598052327-1632135402.1596630124

Burley, B., Adler, D., Chiang, M. J.-Y., Driskill, H., Habel, R., Kelly, P., Kutz, P., Li, Y. K., & Teece, D. (2018). The Design and Evolution of Disney's Hyperion Renderer. *ACM Transactions on Graphics*, *37*(3), 1–22. https://doi.org/10.1145/3182159

Coakley, J. A. (2003). REFLECTANCE AND ALBEDO, SURFACE. In *Encyclopedia of Atmospheric Sciences* (pp. 1914–1923). Elsevier. https://doi.org/10.1016/B0-12-227090-8/00069-5

Coat—Arnold for Maya User Guide—Arnold Renderer. (n.d.). Retrieved 9 August 2020, from https://docs.arnoldrenderer.com/display/A5AFMUG/Coat

Cook, R. L., & Torrance, K. E. (1982). A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics*, *1*(1), 7–24. https://doi.org/10.1145/357290.357293

Dorsey, J., Rushmeier, H., & Sillion, F. X. (2007). *Digital Modeling of Material Appearance* (p. 336). Morgan Kaufmann / Elsevier. https://hal.inria.fr/inria-00510244

DyViTo Project. (2017, October 21). *DyViTo*. https://dyvito.com/dyvitoproject/

Epic Games. (n.d.). *HDRI Backdrop*. Retrieved 12 August 2020, from https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/HDRIBackdrop/index.html

Epic Games. (2020a). *Path Tracer | Unreal Engine Documentation*. https://docs.unrealengine.com/en-US/Engine/Rendering/RayTracing/PathTracer/index.html

Epic Games. (2020b). *Unreal Engine 4* (4.25.0) [Computer software]. Epic Games.

Epic Games. (2020c). *Unreal Engine 4—BRDF* (4.25.0) [Source Code for BRDF.ush]. Epic Games. https://github.com/EpicGames/UnrealEngine

Fleming, R. W., Dror, R. O., & Adelson, E. H. (2003). Real-world illumination and the perception of surface reflectance properties. *Journal of Vision*, *3*(5), 3. https://doi.org/10.1167/3.5.3

Fleming, R. W., Gegenfurtner, K. R., & Nishida, S. (2015). Visual perception of materials: The science of stuff. *Vision Research*, *109*, 123–124. https://doi.org/10.1016/j.visres.2015.01.014

Fuchs, M. (2008). *Advanced Methods for Relightable Scene Representations in Image Space* [Dissertation, Saarland University]. http://scidok.sulb.uni-saarland.de/volltexte/2009/2119/

gallickgunner (username). (2018, April 16). *What is the difference between radiance and irradiance in BRDF* [Discussion Post]. Computer Graphics Stack Exchange. https://computergraphics.stackexchange.com/a/7517

Georgiev, I., Portsmouth, J., Andersson, Z., Herubel, A., King, A., Ogaki, S., & Servant, F. (2019a, August 27). *Autodesk Standard Surface* [Document; White paper]. https://autodesk.github.io/standard-surface/

Georgiev, I., Portsmouth, J., Andersson, Z., Herubel, A., King, A., Ogaki, S., & Servant, F. (2019b). *Standard Surface Shader* [Source Code of reference implementation]. Autodesk. https://github.com/Autodesk/standard-surface

Guarnera, D., Guarnera, G. C., Toscani, M., Glencross, M., Li, B., Hardeberg, J. Y., & Gegenfurtner, K. R. (2018). Perceptually Validated Cross-Renderer Analytical BRDF Parameter Remapping. *IEEE Transactions on Visualization and Computer Graphics*, *26*(6), 2258–2272. https://doi.org/10.1109/TVCG.2018.2886877

Gulbrandsen, O. (2014). *Artist Friendly Metallic Fresnel*. *3*(4), 9.

Guy, R., & Agopian, M. (2020, August 7). *Filament Materials Guide* [Document]. https://google.github.io/filament/Materials.html

Haindl, M., & Filip, J. (2013). *Visual Texture: Accurate Material Appearance Measurement, Representation and Modeling*. Springer Science & Business Media.

Hecht, E. (2018). Optik. In *Optik*. De Gruyter. https://www.degruyter.com/view/title/525251

Heitz, E. (2018). *Sampling the GGX Distribution of Visible Normals*. *7*(4), 13.

Hiranyachattada, T., & Kusirirat, K. (2020). Using mobile augmented reality to enhancing students' conceptual understanding of physically-based rendering in 3d animation. *European Journal of Science and Mathematics Education*, *8*(1), 1–5.

Hullin, M. B., Ihrke, I., Heidrich, W., Weyrich, T., Damberg, G., & Fuchs, M. (2013). *State of the Art in Computational Fabrication and Display of Material Appearance*. 18.

Hunter, R. S. (1937). Methods of determining gloss. *Journal of Research of the National Bureau of Standards*, *18*, 23.

Jones, L. A. (1922). The Gloss Characteristics of Photographic Papers. *JOSA*, *6*(2), 140–161. https://doi.org/10.1364/JOSA.6.000140

Karis, B. (2013, July 25). *Real Shading in Unreal Engine 4* [Conference session updated version 2]. SIGGRAPH 2013 Course: Physically Based Shading in Theory and Practice, Anaheim, California. https://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf

Lafortune, E. P. F., Foo, S.-C., Torrance, K. E., & Greenberg, D. P. (1997). Non-linear approximation of reflectance functions. *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 117–126. https://doi.org/10.1145/258734.258801

Lagarde, S. (2011a, August 17). Adopting a physically based shading model. *Sébastien Lagarde*. https://seblagarde.wordpress.com/2011/08/17/hello-world/

Lagarde, S. (2011b, August 17). Feeding a physically based shading model. *Sébastien Lagarde*. https://seblagarde.wordpress.com/2011/08/17/feeding-a-physical-based-lighting-mode/

Lagarde, S. (2012a, January 8). PI or not to PI in game lighting equation. *Sébastien Lagarde*. https://seblagarde.wordpress.com/2012/01/08/pi-or-not-to-pi-in-game-lighting-equation/

Lagarde, S. (2012b, April 30). DONTNOD specular and glossiness chart. *Sébastien Lagarde*. https://seblagarde.wordpress.com/2012/04/30/dontnod-specular-and-glossiness-chart/

Lagarde, S. (2014, April 14). DONTNOD Physically based rendering chart for Unreal Engine 4. *Sébastien Lagarde*. https://seblagarde.wordpress.com/2014/04/14/dontnod-physically-based-rendering-chart-for-unreal-engine-4/

Lensch, H. P. A. (2004). *Efficient, image-based appearance acquisition of real-world objects* (1. Aufl) [Based on the author's dissertation at the Saarland University]. Cuvillier.

Mipmaps or Nearest Filtering option. (2018, December 11). Blender Developer Talk. https://devtalk.blender.org/t/mipmaps-or-nearest-filtering-option/4344

Moeller, M., & Georgiev, I. (2020, June 12). *Add reference images · Issue #11 · Autodesk/standard-surface* [Issue]. GitHub. https://github.com/Autodesk/standard-surface/issues/11

Montagne, B. (2018, February 27). *FBX File Structure—BlenderWiki*. https://archive.blender.org/wiki/index.php/User:Mont29/Foundation/FBX_File_Structure/

Nicodemus, F. E., Richmond, J. C., Hsia, J. J., Ginsberg, I. W., & Limperis, T. (1977). *Geometrical considerations and nomenclature for reflectance* (NBS MONO 160; 0 ed., p. NBS MONO 160). National Bureau of Standards. https://doi.org/10.6028/NBS.MONO.160

North Carolina Climate Office. (n.d.). *Albedo | North Carolina Climate Office*. Albedo | North Carolina Climate Office. Retrieved 9 August 2020, from https://climate.ncsu.edu/edu/Albedo

Pfeiler, W. (2017). *Optik, Strahlung*. De Gruyter.

Pharr, M., Jakob, W., & Humphreys, G. (2018). *Physically Based Rendering: From Theory to Implementation* (3rd ed.) [Online version]. http://www.pbr-book.org/

Phillips, J. B., Ferwerda, J. A., & Luka, S. (2009). Effects of Image Dynamic Range on Apparent Surface Gloss. *Color and Imaging Conference*, *2009*(1), 193–197.

Phong, B. T. (1975). Illumination for computer generated pictures. *Communications of the ACM*, *18*(6), 311–317. https://doi.org/10.1145/360825.360839

Rusinkiewicz, S., & Marschner, S. (2000). *Measurement I - BRDFs* [Script for the course CS448C: Topics in Computer Graphics]. http://aperture.stanford.edu/courses/cs448c-00-fall/notes/lect02.ps

Schlick, C. (1994). An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*, *13*(3), 233–246. https://doi.org/10.1111/1467-8659.1330233

Schmidt, F., Fleming, R. W., & Valsecchi, M. (2020). Softness and weight from shape: Material properties inferred from local shape features. *Journal of Vision*, *20*(6), 2–2. https://doi.org/10.1167/jov.20.6.2

Schmidt, T.-W., Pellacini, F., Nowrouzezahrai, D., Jarosz, W., & Dachsbacher, C. (2014). State of the Art in Artistic Editing of Appearance, Lighting, and Material. *Eurographics 2014 - State of the Art Reports*. https://doi.org/10.2312/egst.20141041

Serrano, A., Gutierrez, D., Myszkowski, K., Seidel, H.-P., & Masia, B. (2016). An intuitive control space for material appearance. *ACM Transactions on Graphics*, *35*(6), 1–12. https://doi.org/10.1145/2980179.2980242

Smith, B. (1967). Geometrical shadowing of a random rough surface. *IEEE Transactions on Antennas and Propagation*, *15*(5), 668–671. https://doi.org/10.1109/TAP.1967.1138991

Solid Angle S.L. (n.d.-a). *Specular—Arnold for Maya User Guide—Arnold Renderer*. Retrieved 9 August 2020, from https://docs.arnoldrenderer.com/display/A5AFMUG/Specular

Solid Angle S.L. (n.d.-b). *Standard Surface—Arnold for Maya User Guide—Arnold Renderer*. Retrieved 13 August 2020, from https://docs.arnoldrenderer.com/display/A5AFMUG/Standard+Surface

Solid Angle S.L. (n.d.-c). *Subdivision Settings—Arnold for Maya User Guide—Arnold Renderer*. Retrieved 12 August 2020, from https://docs.arnoldrenderer.com/display/A5AFMUG/Subdivision+Settings

Sturm, T., Sousa, M., Thöner, M., & Limper, M. (2016). A unified GLTF/X3D extension to bring physically-based rendering to the web. *Proceedings of the 21st International Conference on Web3D Technology - Web3D '16*, 117–125. https://doi.org/10.1145/2945292.2945293

Unity Technologies. (2014). *Unity - Manual: Material charts* [Manual]. Unity - Manual: Material Charts. https://docs.unity3d.com/Manual/StandardShaderMaterialCharts.html

Vignola, E. (2015, August 31). *[GUIDE] How to install numpy+scipy in Maya Windows 64 bit?* [Guide]. https://forums.autodesk.com/t5/maya-programming/guide-how-to-install-numpy-scipy-in-maya-windows-64-bit/m-p/5796722#M4031

Walter, B., Marschner, S. R., Li, H., & Torrance, K. E. (2007). Microfacet Models for Refraction through Rough Surfaces. *Eurographics Symposium on Rendering*, 12.

Ward, G. J. (1992). Measuring and modeling anisotropic reflection. *ACM SIGGRAPH Computer Graphics*, *26*(2), 265–272. https://doi.org/10.1145/142920.134078

Woo, A., Pearce, A., & Ouellette, M. (1996). It's really not a rendering bug, you see. *IEEE Computer Graphics and Applications*, *16*(5), 21–25. https://doi.org/10.1109/38.536271

Zaal, G. (n.d.). *HDRI Haven*. HDRI Haven. Retrieved 12 August 2020, from https://hdrihaven.com/hdris/

Zinth, W., & Zinth, U. (2009). *Optik: Lichtstrahlen—Wellen—Photonen* (2nd ed.). Oldenbourg. http://d-nb.info/991068505/04

Appendix A: Supplemental material

A.1 Data storage device

The enclosed data storage device contains further material. The folder structure is as follows:

- bittner-bachelorthesis.pdf
- figures
- appendix
  - data
    - 3d-content
    - environment-maps
    - rendered-images
    - figures
  - renderprojects
    - blender
    - maya
    - unreal
  - src

The PDF file corresponds to the printed version of this paper. The folder “figures” contains all figures used in the main part of the paper. All other figures can be found in the folder “appendix”, along with the project data of each rendering software, the source code for the Python and Batch scripts, the scene description in FBX format including the sphere mesh, the used “environment-maps”, all “rendered-images” (following the naming convention: /parameter/environment-name/renderer_environment-name_parameter-value.png), the personal e-mail communication from appendix B and finally the JupyterLab notebooks from appendix A.1.3.

A.1.1 Render projects

There are two scenes in each project that are called *basic-scene* and *directional-light*. The first corresponds to the scene setup with an environment map as light source, while the latter uses the directional light as only light source.

The UE4 project also comprises several other folders with content needed to produce the desired output images:

Folder name	Content
Blueprints	assets containing logic for sequential batch rendering process
FBX	imported assets for the scene setup and geometry for the HDRi Backdrop mesh (see section 3.4)

HDRi	imported environment maps with changes on the import settings
Level	scenes used for rendering
Materials	used materials, including special material with different shading model for the evaluation of *Clearcoat* and a nearly Lambertian material
PostProcessing	neutral sRGB tone mapping operator implemented as post processing material for the camera. However, it is not mandatory to use it as another option for disabling post processing exists by executing a console command

A.1.2 Source Code

There are two different types of scripts, one for executing the batch rendering via command line and another one for the sequential rendering process itself.
The first type comprises two batch scripts per renderer, one for each scene setup (directional light vs. environment map). It triggers a batch rendering via command line in the given renderer with the following syntax

Blender:

```
blender -b scene -P python-script render-engine parameter
environment-name
```

Maya:

```
mayapy python-script scene parameter environment-name
```

where "render-engine" in Blender means either *cycles* or *eevee*, "parameter" corresponds to the parameter name used inside the respective software (e.g. Maya uses *metalness* instead of *Metallic*) and "environment-name" means the string name of any HDR image file without file extension. The directory for the environment maps is defined in the Python script.

There are three Python scripts, two of them for the software Maya and Blender express the implementation of the sequential rendering process inside the renderer. The third script for UE4 is responsible for starting the batch rendering process inside the engine. It must be executed inside the open UE4 editor after the level "basic-scene.umap" has been opened and is neither in Play mode nor Simulation mode. The script will start the rendering process by entering Simulation mode (see section 3.5) and exchanges the environment map. As soon as the blueprint *ParameterRenderer* has sampled one parameter over its entire range and stopped the "game", the Python script restarts the process with another environment map. This is repeated for all environment maps in the project folder "HDRi".

The folder "lib" contains the *numpy* and *scipy* modules that were used in the Maya script. Both must be imported inside the module folder of Maya for a functional program, as described by Vignola (2015).

A.1.3 JupyterLab

The JupyterLab notebook comprises six different Python 3 notebooks:

Python notebook	Content
AverageBrightness	concerns matching unit systems, see section 3.3
CombinedFigures	combines several pre-rendered figures from other notebooks
DifferenceImages	plots the cross-renderer material difference charts [interactive]
LinePlots	plots the scanline plots [interactive]
MaterialCharts	plots the cross-renderer material charts [interactive]
ZipUnzip	convenient method for downloading complete folders

A.2 Additional formulas

A.2.1 Snell's law for ideal refraction

According to Zinth and Zinth (2009, pp. 32–33), Snell's law implies that an incident, transmitted light ray is refracted towards or away from the surface normal. The refraction angle θ_t is the angle between the surface normal that is oriented to the lower hemisphere and the transmitted light ray. The IOR of the medium through which the incident light ray travels is denoted as n_i and the IOR for the medium that the light beam has entered is referred to as n_t. A higher IOR indicates a higher optical density. The relation between the incident and reflection angle as well as both IOR is described as

$$n_i sin(\theta_i) = n_t sin(\theta_t) \qquad \text{A.2.1}$$

and means that the light ray is refracted towards the surface normal (i.e. smaller value for θ_t) when the optically thinner medium enters the optically denser one. Conversely, the light ray is refracted away from the surface normal (i.e. towards the surface itself with $\theta_t \leq 90°$) if the light ray leaves an optically denser medium and enters the optically thinner medium. For example, air ($n \approx 1.0$) is optically thinner than liquid water ($n \approx 1.3$). Hence, a light ray entering the water surface will be refracted towards the surface normal.

A.2.2 Reducing dimensionality of light transport functions

To model a function for light transport, more recent studies have already included 16 dimensions (Haindl & Filip, 2013). However, for simplicity here we refer to the assumption of Rusinkiewicz and Marschner (2000) that light transport on a material surface in its most complex form is a 12D function

$$(x, y, t, \theta, \phi, \lambda)_{in} \rightarrow (x, y, t, \theta, \phi, \lambda)_{out} \qquad \text{A.2.2}$$

that considers six parameters for both incoming and outgoing directions (pp. 1-2). These consist of the position where the light ray enters or leaves the surface (x, y), the moment of interaction t, the incident or outgoing direction defined by two angles (θ, ϕ) and the wavelength of light λ. Because the requirements for storage space and computing power would be too high, the computation or measuring of a material function comprising all twelve dimensions is unfeasible until now. Thus, Rusinkiewicz and Marschner (2000) simplify the model of light transport on a material surface through several assumptions. In a first step, luminescent materials are disregarded, which means that light is immediately deflected ($t_{in} = t_{out}$) when it hits the surface because a delay caused by phosphorescence is not considered. On the other hand, this also implies that the material does not change the wavelength of the incident light ray ($\lambda_{in} = \lambda_{out}$) like fluorescent materials generally do. Additionally, the authors suppose that "the appearance of the surface is constant over time" (p.1) and in the context of this paper, the wavelength of a light beam is neglected as well, as this work does not demand spectrally correct rendering. The resulting function is an 8D function

$$(x, y, \theta, \phi)_{in} \rightarrow (x, y, \theta, \phi)_{out} \qquad \text{A.2.3}$$

that is called the *bidirectional scattering-surface reflectance distribution function* (BSSRDF) and discussed in section 2.3.1.

A.2.3 The Cook-Torrance BRDF

The original BRDF by Cook and Torrance (1982, p. 10) is denoted as

$$f_r = sR_s + dR_d \qquad \text{where} \quad s + d = 1. \qquad \text{A.2.4}$$

The term R_d is the diffuse reflection component, whereas R_s labels the specular reflection. Both are linearly combined, scaled by two factors s and d, whose sum is equal one to ensure energy

conservation. While the diffuse component can be implemented using a Lambertian BRDF (see section 2.4.1), the specular reflection component is defined as

$$R_s = \frac{\mathrm{FDG}}{\pi(\mathrm{N}\cdot\mathrm{L})(\mathrm{N}\cdot\mathrm{V})} \tag{A.2.5}$$

where F is the Fresnel term, D the distribution function of the microfacets and G the shadowing-masking term, which are all discussed in section 2.4.2. N, L and V however denote the normalized vectors for the surface normal, light and viewer. L and V are a different way of expressing the directions ω_i and ω_o. Lastly, the term is normalized by π. However, more recent studies like Walter et al. (2007, pp. 5–6) state that "a factor of 4 in the denominator instead of π" (pp. 5-6) yields better results. Thus, by now the commonly used formulation of the BRDF by Cook and Torrance provided by Walter et al. (2007) is

$$\mathrm{f_r} = \frac{\mathrm{F(i,h_r)G(i,o,h_r)D(h_r)}}{4|\mathrm{i}\cdot\mathrm{n}||\mathrm{o}\cdot\mathrm{n}|} \tag{A.2.6}$$

where i is the incident and o the outgoing normalized light vector, n corresponds to the surface normal vector and $\mathrm{h_r}$ denotes the normalized vector bisecting the incident and outgoing light vectors.

A.2.4 Conversion of "Specular", F_0 and IOR

The following equation is a summary for Burley (2015, p. 13) and Cook and Torrance (1982, p. 16). The IOR is denominated by η. It describes the relationship between the Fresnel reflectance at normal incident angle F_0, the IOR of a material η and the Specular parameter commonly used in principled material models with a metallic workflow:

$$F_0 = \left(\frac{1-\eta}{1+\eta}\right)^2 = 0.08\cdot\mathrm{specular} \tag{A.2.7}$$

A.3 Additional notes

The Mitsubishi Electric Research Laboratories (MERL) database is a freely available resource for measured BRDF data for research and academic use. As Serrano et al. (2016) have demonstrated, new material can be synthesised by "any convex combination of two given BRDFs [...] where non-negativity, energy conservation and reciprocity are preserved" (p. 3).

A method for measuring the similarity of material appearance by deep learning algorithms was proposed by Lagunas et al. (2019) in their paper "A Similarity Measure for Material Appearance".

A.4 Empirical data for lighting settings

	Cycles 1	Cycles 2	Cycles 3	Cycles 4	Cycles 5	Cycles 6
Square Samples	true	true	false	false	false	true
AA samples	16	16	1024	512	128	8
diffuse	4	4	4	8	4	2
glossy	4	4	4	8	4	2
transmission	4	4	4	8	4	2
AO	1	1	1	1	1	2
Mesh light	4	4	4	8	4	2
Subsurface	4	4	4	8	4	2
Volume	4	4	4	8	4	2
Adaptive Sampling	false	false	false	false	false	false
Light Paths						
Total	10	10	10	10	128	10
diffuse	1	1	1	1	128	1
glossy	1	1	1	1	128	1
transparency	10	0	10	10	128	10
transmission	8	8	8	8	128	8
volume	0	0	0	0	128	0
Caustics	true	true	true	true	true	true
samples for environment map	16	16	16	16	16	16
Notes on quality	almost no noise	no difference to Cycles 1	Best trade-off	less noise than Cycles 6, less time than Cycles 3	preset "final" of Blender	corresponds to Arnold settings, fireflies
Total render time with GUI [h]	14:34:51	14:30:37	03:08:53	01:51:47	05:26:05	01:01:32

Table 4: Experimental lighting settings in Cycles, sorted in descending visual quality

Appendix B: Personal communication

With her consent, the most important extracts from the personal e-mail communication with Tiantada Hiranyachattada are given below in chronological order. Please refer to the enclosed data carrier for the complete conversation.

From: Tiantada Hiranyachattadamailto:bmafueng@hotmail.com
Sent: Sunday, May 17, 2020 7:01 AM
To: Franca Bittnermailto:fb076@hdm-stuttgart.de

Subject: Re: Further information about mobile AR application for understanding PBR concepts

Dear Sir or Madam,

[...] these are my details of my application

- we teach PBR by using AR to make students see the result between real object and PBR object created by unity

- this is my reference to do object detection from youtube:
https://www.youtube.com/watch?v=5uyqOX3nSyI

- the example objects will be recognise first from the application before we used teach students (application cannot detect every objects)

- the real object we use were only example objects which were concrete rock, aluminium fork and plastic mug

- the PBR concept we define was a simple concept, we only focus that students must understand 3 main properties of PRB: base color, metalness and roughness

- for the correct PBR parameters value can be reference from Physically Based Materials
https://docs.unrealengine.com/en-us/Engine/Rendering/Materials/PhysicallyBased

- we assess the students understanding by make them do the PBR material in their 3D rendering work, if they can adjust the main parameter correct or not.

[...]

Sincerely,

Tiantada Hiranyachattada

From: Franca Bittnermailto:fb076@hdm-stuttgart.de
Sent: Sunday, May 24, 2020 9:03 PM
To: Tiantada Hiranyachattadamailto:bmafueng@hotmail.com
Subject: Re: Further information about mobile AR application for understanding PBR concepts

Dear Ms Hiranyachattada,

[...]

If I understand correctly, you looked up the desired PBR material parameter values for a specific object online in a list with measured BRDF parameter values and entered the values into the application before handing it over to the students. Is that right? Or did your AR application rather suggest the correct parameter values on its own?

[...]

Kind regards,

Franca Bittner

From: Tiantada Hiranyachattadamailto:bmafueng@hotmail.com
Sent: Wednesday, June 24, 2020 5:26 AM
To: Franca Bittnermailto:fb076@hdm-stuttgart.de
Subject: Re: Further information about mobile AR application for understanding PBR concepts

[...]

the value itself is taken from the Unreal Engine Documentation! Right!

I'm not sure if their any measured parameter from other documentation,

[...]

Tiantada Hiranyachattada