Comparison of material models in modern physically based rendering pipelines

Franca Bittner[1]

Abstract: The appearance of materials results from a complex interaction of light, material properties and the geometric shape of an object. In computer graphics, various models were developed to describe these correlations. Modern rendering pipelines commonly adapt the philosophy of physically based rendering (PBR). This study examines if the reproduction of materials differs across modern PBR tools, and compares the intuitiveness of material design, the quality and range of reproducible materials. A sequential rendering framework was developed to evaluate the visual influences of four selected parameters on material appearance. The rendered images are qualitatively compared based on material charts, scanline plots and difference images. The examined rendering tools mostly yield similar results, with the main differences caused by disparate rendering methods. Still, subtle variations between the tools are noticable, indicating the individual strengths and flaws of each renderer in terms of intuitiveness and physical accuracy.

Keywords: Physically Based Rendering; Material Models; Material Appearance; Rendering Pipeline

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 PBR 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 [Bu12] 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, or the game engines Unity and Unreal Engine 4 (UE4), have adopted the PBR workflow. With software as the application suite Substance by Adobe, there even are tools purely dedicated to material appearance design. 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" [Dy17]. By now, a variety of models for describing

[1] Stuttgart Media University, Nobelstraße 10, 70569 Stuttgart, Germany; since October 2020 at Technische Universität Ilmenau, Ehrenbergstraße 29, 98693 Ilmenau, Germany, franca.bittner@mailbox.org

material appearance exists in terms of PBR. Yet, these models cannot be compared easily. 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 [Gu20]. 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 study aims to compare the different material models in modern rendering software commonly used in digital productions. It is investigated how the material models in current PBR pipelines differ in terms of the intuitiveness of material design and the quality and range of reproducible materials.

2 Related Work

Guarnera et al. [Gu20] 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. Two images of an identical object are compared: both rendered in the same scene, but with different material models. In an iterative process, two parental data sets are combined to generate new possibly fitting parameter sets. This way, the parameter sets of both material models are matched and evoke a nearly identical output image. While that work is more sophisticated than this paper, it does not examine material models commonly used in current digital production workflows in detail. This work is intended to fill this gap.

Burley [Bu12], Karis [Ka13], Lagarde [La11; La12; La14], Unity Technologies [UT14] and the Blender Foundation [BF17] 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 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 [SA]. As reference images are actually in demand for testing different implementations and ensuring look consistency, this work aims to provide a rendering framework to generate such reference material charts across all examined renderers.

Another topic related to this work is the improvement of *material appearance design*. The report of Schmidt et al. [Sc14] deals with alternative ways of editing a virtual scene. The authors define the term “appearance design” and describe several concepts that connect lighting and material editing, as for instance editing the lighting scenario by clicking and dragging a specular highlight of an object [Sc14, pp. 2–5]. In accordance with this work, this study considers the appearance of a scene as an interplay of lighting and surface

materials, or global and local light transport respectively [Sc14, p. 2, 4, 6, 9]. Serrano et al. [Se16] developed an intuitive control space for editing captured bidirectional reflectance-distribution function (BRDF) data and proposed a list of fourteen attributes for material appearance [Se16, p. 6]. However, data-driven BRDFs are less relevant in the context of digital media productions, since in the commonly used tools the representation and editing of those data is not widely supported or feasible due to high memory requirements and expensive calculations. Gulbrandsen [Gu14] depicts an example for mapping unintuitive parameters of a physically plausible model to "artist friendly" [Gu14, 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.

3 Approach

3.1 Evaluation

Four renderers commonly used in the media industry are discussed: Arnold for Maya [Au20], Cycles and Eevee from Blender [BF20a], and the game engine Unreal Engine 4 (UE4) [EG20a]. Both, real-time and offline rendering pipelines are considered, representing different use cases like film production or game development. Arnold and Cycles are based on path tracing and thus considered as offline renderers. Although UE4 is capable of path tracing as well, only the real-time rasterization rendering pipeline is evaluated in this paper. Likewise, Eeve is a real-time rasterization renderer.

For the examination, several images depicting an object with different material properties in a certain lighting condition are generated. One parameter at a time is incremented with a specific step size to vary material appearance. A new image is rendered for each parameter change. 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 previous related work [Bu12, p. 13][La11; La12; La14][UT14].

The script for this process was implemented individually for each rendering software in the scripting language Python (and additionally with Blueprints in UE4). The set of rendered images is then composed in a row for each renderer and plotted against the other rendering tools in a Jupyter Notebook. Moreover, difference images and scanline plots are generated. The source code is provided on the project website[2].

Four material parameters are examined: *Roughness*, *Metallic*, *Specular* and *Clearcoat*. While the latter is a rather advanced appearance feature, the other three parameters are the well-known basic parameters to influence materials in a PBR workflow. The used shader models are "Principled BSDF" in Blender, "Standard Surface" shader in Arnold

[2] https://www.hdm-stuttgart.de/ifg/projects/2021/Bittner_2021_CMM

and “Default Lit” and “Clear Coat” model in UE4. In some cases, the names of the parameters vary slightly among the tools. Here, the parameter with the more similar visual impact is selected. Arnold distinguishes between *specularRoughness* and *diffuseRoughness*. *specularRoughness* behaves analogous to *Roughness* in the other rendering tools, as both control the sharpness of the specular highlight. Also, *Metalness* in Arnold is used to evaluate the parameter *Metallic* in this paper, whereas *Coat* corresponds to *Clearcoat*.

Since a comprehensive quantitative investigation was beyond the scope of this paper, a qualitative comparison method is chosen. The images are rendered at a resolution of 500x500 pixels and compared by the author 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.

3.2 Test environment

Four main aspects influence the final object appearance apart from material properties: camera, light, geometry, and the algorithm for emulating light transport [Sc14, p. 2]. As the latter differs in each rendering tool, certain visual differences will be inevitable. The implementation of the light transport is not manipulated, because this would not represent the general working environment of a 3D artist.

For this paper, two different lighting scenarios were chosen. The first one consists of a natural lighting situation, which is implemented using an environment map and image-based lighting (IBL). 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, e. g. in architecture design or virtual film production. IBL strongly depends 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, the second lighting scenario is a very unnatural one, composed of only one directional light, to provide images that are generated with a more basic lighting feature. Directional lights are commonly controllable by two parameters, the power of the light source and the rotation or direction of light. This simplified lighting scenario is more likely to be implemented similarly in each software. Hence, the lighting scenario may be less error-prone with less causes for mismatch.

Four environment maps were chosen to evaluate different material appearance features. All high dynamic range images (HDRIs) are freely available and were taken from the website HDRIHaven[3][Za]. The chosen lighting scenarios range from high to medium dynamic range, as depicted in Table 1. Contrary to my initial expectations, the night scene “Moonless Golf” did not provide added value for recognizing the interplay of lighting and material appearance. The lighting scenario was still useful for determining sufficient lighting settings for the rendering process, as dark scenes are challenging due to the small amount of light.

[3] https://hdrihaven.com/hdris/

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

Tab. 1: Properties of used environment maps

The shape of an object can either be concave or convex and determines the observable material characteristics. For simplicity, a convex sphere is used. 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 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, but may provide additional information about material appearances features. The resolution of a surface mesh is critical as well. A too low resolution of the polygonal mesh may result in artifacts like the *terminator problem* [WPO96, p. 22]. 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.

4 Results

The results are described from two perspectives: First, the impact of a parameter within the same renderer allows for conclusions about the intuitiveness of material design. If the parameter causes different effects than expected, the descriptor may be classified as unintuitive. Secondly, the parameter impact is compared across all rendering tools to evaluate the impact of different material models on material appearance.

Two types of material charts depict the influence of a material parameter. The general material chart contrasts the rendered images unmodified. The difference image material charts oppose the difference image between two rendered images, where white areas remark unchanged areas, a red tone indicates an increase, and blue colouration a decrease in brightness. The colormap is normalized to the maximum negative and positive values.

The complete collection of figures and material charts is provided as supplementary material on the project website[4].

4.1 Roughness

The parameter *Roughness* has a great impact on the appearance of a material. Many characteristic features like specular highlights or reflections can only be observed if the

[4] https://www.hdm-stuttgart.de/ifg/projects/2021/Bittner_2021_CMM

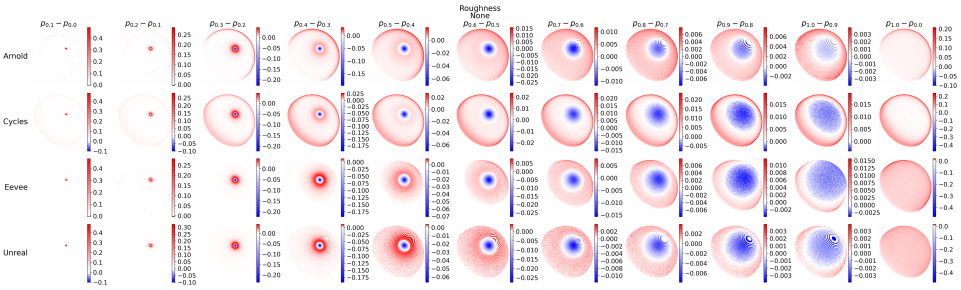

Fig. 1: Cross-renderer material difference chart for “Roughness” without environment map. The last image shows the difference image between value 0.0 and 1.0

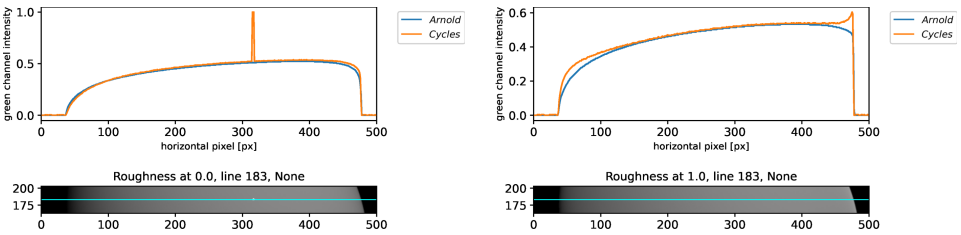

Fig. 2: Brightness plot for “Roughness” 0.0 (left) vs. 1.0 (right) in Arnold and Cycles. The image below shows the plotted image line 183 (cyan) in the context of the total image, rendered in Cycles.

surface is smooth. Lower values of *Roughness* correspond to smooth surfaces, while higher values correspond to a rougher surface. The behaviour of specular highlights in all rendering tools corresponds to the laws of physics for specular and diffuse reflection. Rough surfaces scatter light in many directions, resulting in a wider and less bright highlight, and vice versa. Hence, the general impact of the parameter *Roughness* behaves very intuitively.

When varying the parameter *Roughness*, the results are quite different across each renderer. 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 Fig. 1, this appearance feature is not present in the two rasterizers. In a detailed look at the differences between Arnold and Cycles, Cycles considers retro-reflection, i. e. 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). Fig. 2 contrasts 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, i. e. the edge of the sphere. The peak in brightness at grazing angles is even higher than the general diffuse colour of the sphere and might be interpreted as retro-reflection. Interestingly,

only Cycles appears to model this phenomenon correctly in the lighting situation with one directional light.

4.2 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. 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 in Fig. 3. The final appearance of a metallic material thus depends more on its environment than a dielectric material would. In general, the parameter *Metallic* causes expected intuitive results, since we are used to metallic surfaces being more reflective.

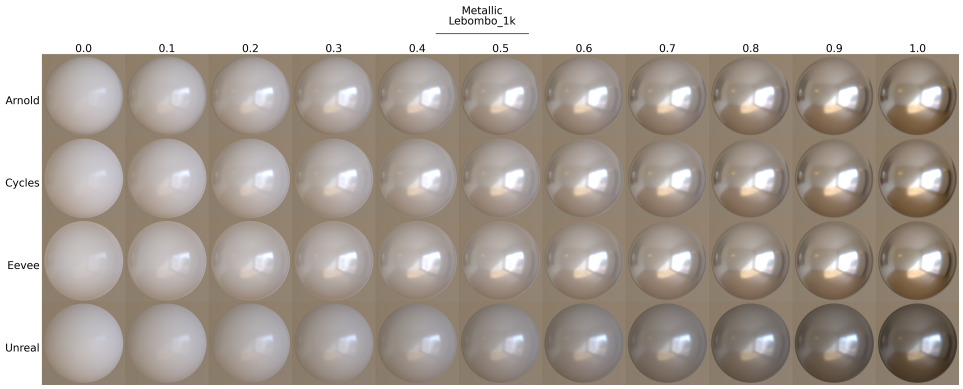

Fig. 3: Cross-renderer material chart for “Metallic” in environment “Lebombo”

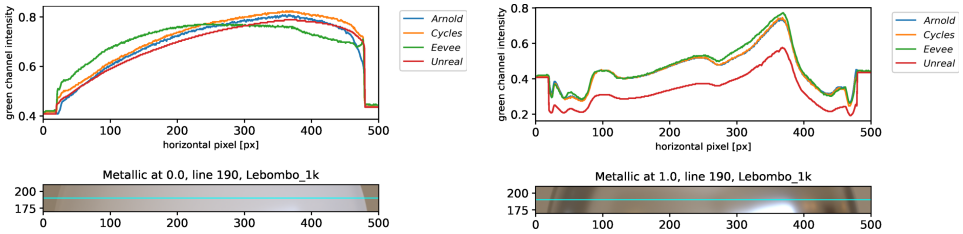

Fig. 4: Intensities for dielectric vs. metallic surface across all renderers with noticable drop-off in UE4. The image below shows the plotted image line 190 (cyan) in the context of the total image, rendered in Cycles.

The most noticeable difference for the parameter *Metallic* across all renderers was the significantly darker appearance of metallic materials in UE4. Fig. 3 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, Fig. 4 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. Nevertheless, there is a significant decrease in intensity of about 0.1 for metallic materials in UE4 compared with the other renderers in this environment. The other lighting settings "sunflowers" and "moonless_golf" produce similar results, only the environment map "colorful_studio" evoked consistent intensity values across all renderers. Unfortunately, I could not find a sufficient explanation for the deviating test results of Unreal in the mentioned light situations.

4.3 Specular

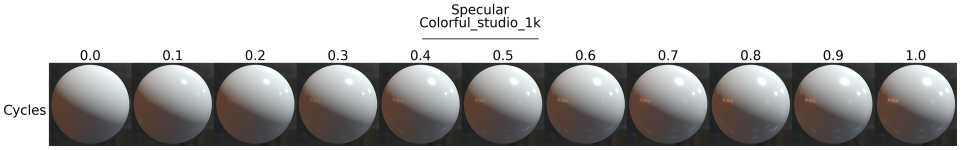

Fig. 5: Varying "Specular" in Cycles in environment "Colorful Studio"

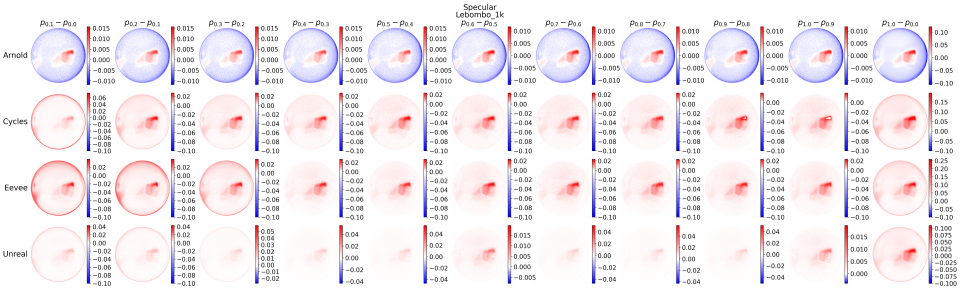

Fig. 6: Cross-renderer material difference chart for "Specular" in environment "Lebombo" . The last image shows the difference image between value 0.0 and 1.0.

The *Specular* value influences the intensity of all specular reflections (see Fig. 5). The parameter only has any impact if the material is non-metallic (i.e. dielectric). However, I claim that this parameter concept is not very intuitive and the term "specular" not self-explanatory. In fact, the reflection of pure metallic surfaces only consists of a specular component (i. e. no diffuse reflection), so it is misleading that the *Specular* parameter has no effect on metals.

Its visual effect can easily be mistaken for *Roughness*, however the parameter *Specular* does not affect the fuzziness of the specular highlight. Instead, it acts like a multiplier for the intensity of the highlight. According to the reference implementation of the "Autodesk Standard Surface" shader [Ge19, pp. l. 126–129], the *Specular* parameter indeed acts as a multiplier for the specular reflection intensity. The index of refraction (IOR) for calculating

the Fresnel term is exposed as a separate parameter in Arnold. In contrast, the *Specular* parameter in Blender and UE4 controls the IOR of the material directly. A conversion between *Specular*, F_0 and the IOR is provided by Burley [Bu15], based on the formulation for F_0 by Cook and Torrance [CT82], and adopted by the mentioned renderers [BF20b, pp. l. 101][EG20b, pp. ll. 76–97].

The physical term IOR is less intuitive than the parameter *Specular* [Bu15]. 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. However, one could argue that the *Specular* parameter in Arnold allows for more traditional control over the amount of specular reflections. The specular highlight acts as separate layer and 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. The greatest issue in terms of intuitiveness is the non-consistent usage of the descriptor *Specular* across the examined tools. The parameters share the same descriptor but affect material features in a fundamental different manner. This semantic mismatch can easily get confusing for the user.

The different effects of *Specular* across all renderers is also observable in the material difference chart (see Fig. 6). While the image intensity at grazing angles linearly decreases in Arnold (i. e. less reflective), it increases non-linearly for the renderers UE4, Cycles and Eevee . It can be assumed that this is due to different calculations of the Fresnel term, since the grazing shadow of smooth surfaces can be derived from the Fresnel equations [Bu12, p. 6]. In the inner area of the sphere, all renderers behave linearly. Overall, the visual disparities among the renderers for *Specular* in the examined lighting situations are minimal and mostly noticable towards grazing angles.

4.4 Clearcoat

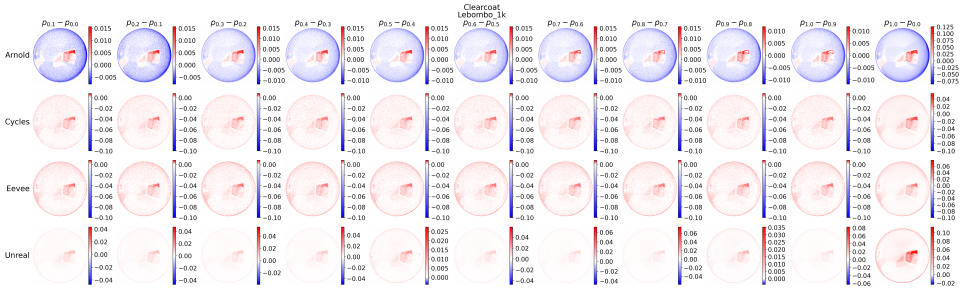

Fig. 7: Cross-renderer material difference chart for “Clearcoat” in environment “Lebombo”

The impact of *Clearcoat* is very subtle in the examined lighting situations and best observable with the renderer Arnold viewed in the lighting situation “Lebombo” or “Colorful Studio”. The effect can 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. The

underlying blurry highlight typical for rough surfaces is still perceivable, although there is a specular layer on top of it. 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. 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.

The difference image material charts in Fig. 7 revealed a deeper insight of the behaviour of this parameter. 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. The increment of the parameter adds a grazing shadow in Arnold, while the other renderings show an increased reflectivity towards grazing angles instead. This again suggests that the Fresnel term for modelling the clearcoat layer is implemented differently in the material models. 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.

5 Discussion

My findings show that the examined rendering software has mainly adopted the same underlying PBR workflow. The main differences occurr 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 divergent scene appearance in each tool.

Although all listed disparities are subtle and mostly not perceptible when working within one rendering tool, they 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.

6 Conclusion

This study 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 arise from different rendering mechanisms rather than the underlying material model. 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 be dedicated for working within a rendering tool.

References

[Au20] Autodesk: Maya, comp. software, version 2020, 2020.

[BF17] Blender Foundation: Principled BSDF — Blender Manual, material charts added by Alexander Gavrilov (https://developer.blender.org/rBM3686), July 2017, URL: https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/principled.html, visited on: 08/09/2020.

[BF20a] Blender Foundation: Blender, comp. software, version 2.83.1, 2020.

[BF20b] Blender Foundation: Cycles - node_principled_bsdf.osl, Medium: Source Code, Mar. 2020, URL: https://github.com/blender/blender/blob/master/intern/cycles/kernel/shaders/node_principled_bsdf.osl, visited on: 08/07/2020.

[Bu12] Burley, B.: Physically Based Shading at Disney, Conference session updated version 3, Aug. 2012, URL: http://disney-animation.s3.amazonaws.com/uploads/production/publication_asset/48/asset/s2012_pbs_disney_brdf_notes_v3.pdf, visited on: 06/13/2020.

[Bu15] Burley, B.: Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering, course notes, Los Angeles, California, Aug. 2015, URL: https://blog.selfshadow.com/publications/s2015-shading-course/burley/s2015_pbs_disney_bsdf_notes.pdf, visited on: 08/22/2020.

[CT82] Cook, R. L.; Torrance, K. E.: A Reflectance Model for Computer Graphics. ACM Transactions on Graphics 1/1, pp. 7–24, Jan. 1982, ISSN: 0730-0301.

[Dy17] DyViTo Project, Oct. 2017, URL: https://dyvito.com/dyvitoproject/, visited on: 08/04/2020.

[EG20a] Epic Games: Unreal Engine 4, comp. software, version 4.25.0, May 2020.

[EG20b] Epic Games: Unreal Engine 4 - BRDF, Medium: Source Code for BRDF.ush, July 2020, URL: https://github.com/EpicGames/UnrealEngine, visited on: 08/07/2020.

[Ge19] Georgiev, I.; Portsmouth, J.; Andersson, Z.; Herubel, A.; King, A.; Ogaki, S.; Servant, F.: Standard Surface Shader, Medium: Source Code of reference implementation, Sept. 2019, URL: https://github.com/Autodesk/standard-surface, visited on: 08/07/2020.

[Gu14] Gulbrandsen, O.: Artist Friendly Metallic Fresnel. Journal of Computer Graphics Techniques 3/4, p. 9, 2014, ISSN: 2331-7418.

[Gu20] Guarnera, D.; Guarnera, G. C.; Toscani, M.; Glencross, M.; Li, B.; Hardeberg, J. Y.; Gegenfurtner, K. R.: Perceptually Validated Cross-Renderer Analytical BRDF Parameter Remapping. IEEE Transactions on Visualization and Computer Graphics 26/6, pp. 2258–2272, June 2020, ISSN: 1941-0506.

[Ka13] Karis, B.: Real Shading in Unreal Engine 4, Conference session updated version 2, July 2013, URL: https://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf, visited on: 08/12/2020.

[La11] Lagarde, S.: Feeding a physically based shading model, Aug. 2011, URL: https://seblagarde.wordpress.com/2011/08/17/feeding-a-physical-based-lighting-mode/, visited on: 08/12/2020.

[La12] Lagarde, S.: DONTNOD specular and glossiness chart, Apr. 2012, URL: https://seblagarde.wordpress.com/2012/04/30/dontnod-specular-and-glossiness-chart/, visited on: 08/12/2020.

[La14] Lagarde, S.: DONTNOD Physically based rendering chart for Unreal Engine 4, Apr. 2014, URL: https://seblagarde.wordpress.com/2014/04/14/dontnod-physically-based-rendering-chart-for-unreal-engine-4/, visited on: 08/12/2020.

[SA] Solid Angle S.L.: Standard Surface - Arnold for Maya User Guide - Arnold Renderer, URL: https://docs.arnoldrenderer.com/display/A5AFMUG/Standard+Surface, visited on: 08/13/2020.

[Sc14] Schmidt, T.-W.; Pellacini, F.; Nowrouzezahrai, D.; Jarosz, W.; Dachsbacher, C.: State of the Art in Artistic Editing of Appearance, Lighting, and Material. In (Lefebvre, S.; Spagnuolo, M., eds.): Eurographics 2014 - State of the Art Reports. The Eurographics Association, 2014.

[Se16] Serrano, A.; Gutierrez, D.; Myszkowski, K.; Seidel, H.-P.; Masia, B.: An Intuitive Control Space for Material Appearance. ACM Trans. Graph. 35/6, Nov. 2016, ISSN: 0730-0301, URL: https://doi.org/10.1145/2980179.2980242.

[UT14] Unity Technologies: Unity - Manual: Material charts, 2014, URL: https://docs.unity3d.com/Manual/StandardShaderMaterialCharts.html, visited on: 08/12/2020.

[WPO96] Woo, A.; Pearce, A.; Ouellette, M.: It's really not a rendering bug, you see. IEEE Computer Graphics and Applications 16/5, pp. 21–25, Sept. 1996, ISSN: 1558-1756.

[Za] Zaal, G.: HDRI Haven, URL: https://hdrihaven.com/hdris/, visited on: 08/11/2020.