Utilizing Measured Reflectance for Real-time Rendering in Game Engines

Lukáš Cezner*
Supervised by: Vlastimil Havran†

Department of Computer Graphics and Interaction
Czech Technical University in Prague
Prague / Czech Republic

Abstract

Having an appropriate reflectance model is a crucial part of achieving realistic rendering. Currently, the vast majority of renderers rely on physically-based analytic models with a few parameters. In this paper, we consider another approach and explore the possibility of using measured reflectance data to render 3D objects covered with real-world materials in real-time graphics. Specifically, we created an implementation for two major game engines, Unity and Unreal Engine 5, and compared it with the analytic model. For these purposes, more than 200 samples of materials were measured, and an application was developed to process the measured data.

Keywords: Bidirectional reflectance distribution function, real-time rendering, game engine, direct lighting

1 Introduction

One of the main tasks in computer graphics is to compute realistic images. This task can be achieved using the rendering equation, which incorporates a bidirectional reflectance distribution function (BRDF, see Section 1.2). This function is often described as an analytic model, a polynomial with certain parameters.

In the following sections, we present a different method for expressing a BRDF, which involves an interpolation of values obtained from real-world materials. We describe a workflow that consists of measuring the reflectance of a material, processing the measured data, and rendering the surface with these BRDF values.

1.1 Spherical coordinate system

The spherical coordinate system represents a vector $\vec{v} = (x, y, z)$ using two angles θ, ϕ and a radial distance $r = ||\vec{v}||$. The angle θ is characterized as the angle between the vector $\vec{v}$ and the basis vector z, while ϕ denotes the angle between the basis vector x and the projection of the vector $\vec{v}$ on the xy plane. The conversion between a Cartesian and a spherical coordinate system can be described as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r \cdot \sin\theta \cdot \cos\phi \\ r \cdot \sin\theta \cdot \sin\phi \\ r \cdot \cos\theta \end{bmatrix}, \begin{bmatrix} \theta \\ \phi \\ r \end{bmatrix} = \begin{bmatrix} \arccos\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right) \\ \arctan\left(\frac{x}{y}\right) \\ \sqrt{x^2+y^2+z^2} \end{bmatrix}. \tag{1}$$

On many occasions, we used the spherical coordinate system to describe the direction $\vec{\omega}$ (a unit vector, $r = 1$). An illustration of this situation is shown in Figure 1.

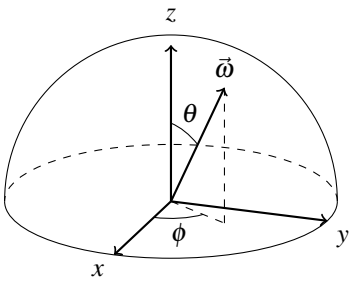


Figure 1: A direction vector $\vec{\omega}$ in spherical and Cartesian coordinate system.

1.2 Bidirectional Reflectance Distribution Function

The bidirectional reflectance distribution function (BRDF) is a mathematical representation of how light is reflected from the surface of an opaque material. For a specific wavelength of light, this is a function of two direction vectors $\vec{\omega}_{in}, \vec{\omega}_{out}$ in spherical coordinates ($\vec{\omega}_{in} = (\theta_{in}, \phi_{in}), \vec{\omega}_{out} = (\theta_{out}, \phi_{out})$) that represent the incoming (light) direction $\vec{l}$ and the outgoing (view) direction $\vec{v}$. The value of the BRDF is determined by the ratio of the reflected radiance L_{out} in direction $\vec{\omega}_{out}$ to the incoming irradiance E_{in} from direction $\vec{\omega}_{in}$ [14]:

$$f(\vec{\omega}_{in}, \vec{\omega}_{out}) = \frac{dL_{out}(\vec{\omega}_{out})}{dE_{in}(\vec{\omega}_{in})} = \frac{dL_{out}(\vec{\omega}_{out})}{dL_{in}(\vec{\omega}_{in}) \cdot \cos\theta_{in}} \quad [\mathrm{sr}^{-1}], \tag{2}$$

Physically plausible BRDFs must obey two restrictions: Helmholtz reciprocity and energy conservation.

*cezneluk@fel.cvut.cz
†havran@fel.cvut.cz

Helmholtz reciprocity defines the relationship between a light ray and its corresponding reverse ray that the value of the BRDF function must be the same:

$$\forall \vec{\omega}_{in}, \vec{\omega}_{out} \in \Omega : f(\vec{\omega}_{in}, \vec{\omega}_{out}) = f(\vec{\omega}_{out}, \vec{\omega}_{in}). \quad (3)$$

Energy conservation is a requirement that the overall outgoing energy cannot exceed the incoming energy:

$$\forall \vec{\omega}_{in} \in \Omega : \int_{\Omega} f(\vec{\omega}_{in}, \vec{\omega}_{out}) \cdot \cos\theta_{out} \, d\vec{\omega}_{out} \leq 1. \quad (4)$$

1.3 Rendering equation

The rendering equation [8] describes the total outgoing radiance in direction $\vec{\omega}_{out}$ at a specific point on the surface:

$$L_{out}(\vec{\omega}_{out}) = L_{emit}(\vec{\omega}_{out}) + \int_{\Omega} f(\vec{\omega}_{in}, \vec{\omega}_{out}) \cdot L_{in}(\vec{\omega}_{in}) \cdot \cos\theta \, d\vec{\omega}_{in}. \quad (5)$$

where L_{emit}, L_{in} are the emitted and the incoming radiance. Due to the integral over a hemisphere and an incoming radiance in it (resulting in the need of a recursive evaluation of the equation), the exact value of this function is computationally demanding. Therefore, this function must be approximated, even more so with real-time rendering.

2 Related work

In this section, we will describe two fields of study that are related to this work: BRDF data sets and analytical representation of BRDF.

2.1 BRDF data sets

The most well-known data set of measured BRDFs is the MERL BRDF database, produced by Matusik et al. [11] They used a spherically homogeneous sample, a stationary camera, and a light on a turntable. MERL database contains 100 different isotropic materials, each of which consists of 1,458,000 samples.

One of the newer data sets worth mentioning was produced by Dupuy and Jakob [4]. They invented an adaptive parameterization, using which they can measure and store only important parts of the BRDF 4D domain. By employing this method, their database currently includes 62 various materials.

2.2 Analytical models of BRDF

Currently, the vast majority of renderers rely on analytic models. One of the fundamental models is the Phong illumination model [15], which lacks both energy conservation and Helmholtz reciprocity. These problems were later solved by Lafortune and Willems [10], who represent the model as:

$$f(\vec{\omega}_{in}, \vec{\omega}_{out}) = \frac{k_d}{\pi} + k_s \cdot \frac{n+2}{2\pi} \cdot (\max\{\vec{v} \cdot \vec{r}, 0\})^n, \quad (6)$$

where $k_d \in [0,1]$, $k_s \in [0,1]$ ($k_d + k_s = 1$) are coefficients of the diffuse and specular part, $\vec{r}$ is a vector of ideal reflection of $\vec{l}$, and $n \in [0,\infty)$ is a parameter that defines the shininess of the material.

Today, more complex models are used. A frequently used model is, for example, the model developed by Walter et al. [21] It is based on the Cook-Torrance model, which, unlike Walter's model, does not satisfy energy conservation.

$$f(\vec{\omega}_{in}, \vec{\omega}_{out}) = k_d \cdot f_d(\vec{\omega}_{in}, \vec{\omega}_{out}, \lambda) + k_s \cdot f_s(\vec{\omega}_{in}, \vec{\omega}_{out}) \quad (7)$$

The specular part of the model is based on microfacet theory and is decomposed into three parts: Fresnel function F, geometric attenuation G, and distribution function D:

$$f_s(\vec{\omega}_{in}, \vec{\omega}_{out}) = \frac{F \cdot G \cdot D}{4 \cdot (\vec{n} \cdot \vec{l}) \cdot (\vec{n} \cdot \vec{v})}, \quad (8)$$

where $\vec{n}$ is a normal of the surface. For Fresnel function F in real-time graphics, the Schlick approximation [16] is used. Geometric attenuation G is influenced by the chosen distribution function F, for which several variants have been introduced. The most used is GGX:

$$D = \frac{\alpha^2 \cdot \max\{0, \vec{h} \cdot \vec{n}\}}{\pi \cdot \cos^4\theta_h (\alpha^2 + \tan^2\theta_h)^2}, \quad (9)$$

$$G \approx G_1(\vec{v}) \cdot G_1(\vec{l}), \quad (10)$$

$$G_1(\vec{x}) = \max\left\{0, \frac{\vec{x} \cdot \vec{n}}{\vec{x} \cdot \vec{h}}\right\} \cdot \frac{2}{1 + \sqrt{1 + \alpha^2 \cdot \tan^2\theta_x}}, \quad (11)$$

where $\vec{h}$ is a half vector ($\vec{h} = \frac{\vec{l}+\vec{v}}{||\vec{l}+\vec{v}||}$), θ_h is an angle between $\vec{h}$ and normal $\vec{n}$, and θ_x is an angle between $\vec{x}$ and normal $\vec{n}$. The parameter α defines the roughness of the material.

3 BRDF Measurements

Acquiring measured BRDF data is the first step in the workflow. We used MiniDiff v2 [17] (shown in Figure 2), a portable contact scatterometer created by Synopsys (previously LightTec). It supports the measurement of BRDF for isotropic materials at four angles of incidence for light sources: $\theta_{in} \in \{0^\circ, 20^\circ, 40^\circ, 60^\circ\}$. For each angle of light, it produces measurements of the RGB reflectance values in the range of $\phi_{out} \in [0^\circ, 360^\circ)$ and $\theta_{out} \in [0^\circ, 75^\circ]$ with a precision of 1°. Therefore, the measurement of a single sample consists of 324,000 BRDF values.

A limited number of material samples can be properly measured as a result of the construction of this instrument. Any sample that is not solid, is not homogeneous, is squashy, has bumps (such as plaster), contains tiny holes (like most fabrics), or is partially transparent (like certain types of plastic) will produce invalid results.

With these constraints, 216 measurements of materials were produced, mainly papers and swatches, but also

Figure 2: MiniDiff v2 with calibration samples.

metal, plastics, felt wool, and stiff foam. Some samples are wood (plywood, chipboard, planed wood), cloth, and leather. Anisotropic materials were measured for two ϕ_{in} angles of light that are approximately perpendicular and stored as independent measurements. The miniatures of all material samples are shown in Figure 9.

4 Processing

During the processing stage, two primary tasks need to be performed: extrapolation and export to the look-up table (LUT) image data. For this reason, we have created an application that also enables us to visualize data and verify its validity. This application was developed as modular and general as possible: internally it works with measurements as a point cloud, and loading data from a new file format can be easily added.

4.1 Extrapolation

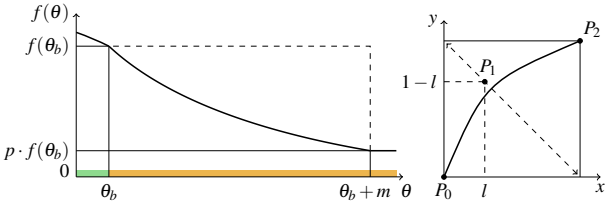
BRDF data for grazing angles ($\theta_{out} > 75°$) are unavailable, so we must extrapolate them from the measured range. Specifically, for each 3D texture slice corresponding to a particular θ_{in}, the values $f(\theta_e)$ are calculated from the value $f(\theta_b)$ of the nearest known measurement (in our scenario $\theta_b = 75°$) as an interpolation of the scale of $f(\theta_b)$ from 1 to the parameter $r \in [0,\infty)$:

$$\begin{aligned} f(\theta_e) &= f(\theta_b) \cdot ((1-\alpha) + \alpha \cdot p), \\ \alpha &= b\left(\min\left\{\frac{\theta - \theta_b}{m}, 1\right\}, l\right), \alpha \in [0,1], \end{aligned} \tag{12}$$

where m represents the maximum expected distance between the known measurement at θ_b and the calculated value at θ_e (in our scenario $m = 90° - 75° = 15°$). Figure 3a shows an example of extrapolated BRDF values. The function $y = b(x,l)$ can be described as finding the coordinate y of a point with the specific coordinate x on the restricted quadratic Bezier curve, illustrated in Figure 3b, specified by $l \in [0,1]$. This curve has a fixed starting point $\vec{P}_0 = (0,0)$, an ending point $\vec{P}_2 = (1,1)$, and a parameterized control point $\vec{P}_1 = (l, 1-l)$:

$$y = b(x,l) \text{ if } \exists t : \begin{bmatrix} x \\ y \end{bmatrix} = 2 \cdot (1-t) \cdot t \cdot \begin{bmatrix} l \\ 1-l \end{bmatrix} + t^2. \tag{13}$$

The equation for this function is solved by testing the roots of the variable t.



(a) Chart of BRDF values by θ (extrapolation area is orange).

(b) Used Bezier curve.

Figure 3: Extrapolation of BRDF data with the parameters $p = 0.2$ and $l = 0.3$.

This extrapolation was developed to be easily computed. If parameter $p = 1$, this extrapolation produces the same results as clamping values outside the measurement region, which is preferable for materials that exhibit mainly diffuse reflection, because it is expected that the BRDF value will not depend much on the outgoing direction $\vec{\omega}_{out}$. For glossy materials, it is recommended to set the parameter $p < 1$, because decreasing the value of the BRDF with increasing θ_{out} (consequently increasing the distance from the direction of ideal reflection) will roughly estimate the shape of the reflection lobe.

4.2 Export to LUT image

During rendering, the isotropic BRDF data are stored as a 3D texture constructed from several 2D textures, slices with the fixed third texture coordinate. The isotropic BRDF has three independent parameters and has $\phi_{in} = 0$. Therefore, each slice represents the BRDF values for a specific θ_{in} in an equirectangular projection with a spherical coordinate system $\vec{\omega}_d = (\theta_d, \phi_d)$ aligned with the direction of specular reflection, i.e. the direction $\theta_d = 0°$, $\phi_d = 0°$ corresponds to $\theta_{out} = \theta_{in}, \phi_{out} = \phi_{in} + 180°$. Slices are arranged in row-major order in an image file with ascending θ_{in} in a single composite image.

For rendering in Unreal Engine, it is necessary to generate a single texture (referred to as an atlas) that merges all measured BRDF samples being used (see Section 5.3). An example of an atlas is illustrated in Figure 5. The layout of an atlas is the same as that for a single BRDF, but instead of the slice itself, the atlas contains a regular grid. Each cell in the grid corresponds to a slice of a single BRDF texture with the same θ_{in}. Cells within the grid are organized similarly to slices in 3D texture, following a row-major order.

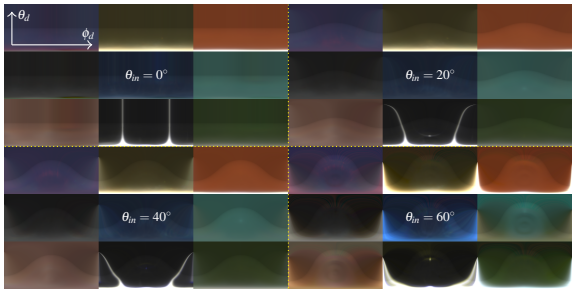


Figure 5: An example of an exported atlas image for 9 different materials.

4.3 Previewing BRDF data

The developed application supports two ways of previewing the measured BRDF data: as a reflection lobe for a specific angle of light $\vec{\omega}_{in}$ and rendering of a 3D object illuminated by an environment map.

The reflection lobe, illustrated in Figure 4a, can be described as a deformed sphere where the distance between the points from the origin corresponds to a specific BRDF value in a particular direction. This visualization method is beneficial for verifying the validity of specific values and gaining insight into the general form of a BRDF.

In contrast, rendering of a 3D object illuminated by an environment map, shown in Figure 4b, can be practical for comparing the real-world and rendered versions of the material. It is computed as a numerical integration of the rendering equation over a hemisphere, where the input radiance is sampled from the environment map in a particular direction.

5 Rendering in game engines

We implemented a shader for two major game engines, Unity and Unreal Engine 5, which compute direct lighting using a 3D LUT image created in the previous step.

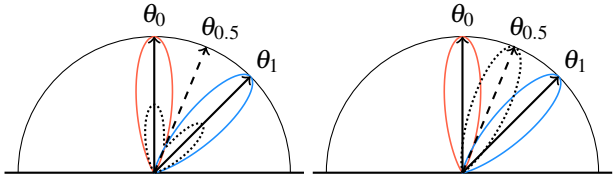
5.1 BRDF evaluation

In practical terms, we assume that the initial slice of the texture corresponds to $\theta_{in,0} = 0$, with each subsequent slice increasing its angle linearly (i.e. $\theta_{in,i+1} = \theta_{in,i} + \Delta$; $\Delta \in \mathbb{R}^+$, specifically in our case $\Delta = 20°$). Consequently, the value of a BRDF point in a normalized texture coordinate can be represented as:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \frac{\phi_d}{2\pi} \\ \frac{\theta_d}{\pi} \\ \frac{\theta_{in}}{\theta_{in,m}} \end{bmatrix}, \tag{14}$$

where $\theta_{in,m}$ is an angle of the last 2D slice in the 3D texture.

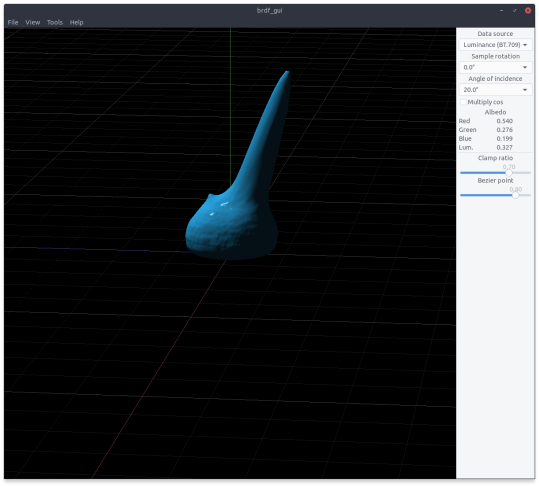
To read the BRDF data from the texture, native trilinear interpolation is used. In the selected coordinate system described in Section 4.2, two points with the same (u,v) in a different 3D texture slice (different texture coordinate w) describe the change of the lobe around the specular reflection, which is more valuable information than the linear interpolation of the BRDF values with fixed $\vec{\omega}_{out}$, because it is expected that the main difference between the 3D texture slices will be in the specular part of the BRDF. For example, in Figure 6, interpolation with fixed $\vec{\omega}_{out}$ produces two smaller lobes, which does not represent the correct behavior.



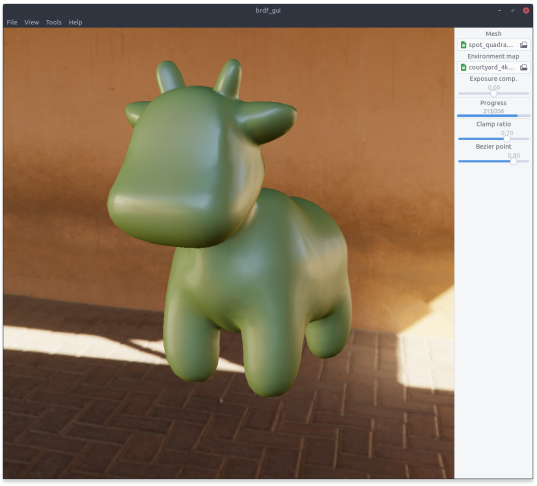
(a) Interpolation between slices parameterized with $\vec{\omega}_{out}$.

(b) Interpolation between slices parameterized with $\vec{\omega}_d$.

Figure 6: Comparing of interpolations of BRDF values (dotted line) of the cosine lobe $(\vec{r} \cdot \vec{v})^{20}$ ($\vec{r}$ is a direction of ideal reflection) between slices with specified θ_{in}.



(a) A reflection lobe.



(b) 3D object illuminated by an environment map.

Figure 4: Implemented methods for previewing BRDF data.

This approach is inappropriate for materials that exhibit substantial retroreflective properties, but such cases are not within the scope of this work.

BRDF is evaluated in a tangent space of the specified point on a 3D object surface. This tangent space, illustrated in Figure 8, is derived from the normal vector $\vec{n}$ and the tangent vector $\vec{t}$, which is formed by projecting the incoming direction $\vec{\omega}_{in}$ onto a plane perpendicular to the normal $\vec{n}$.

To evaluate the BRDF values from the 3D texture, it is necessary to transform the view direction $\vec{\omega}_{out}$ into $\vec{\omega}_d$ in the texture coordinate system. This can be done through the rotation matrix from normal $\vec{n}$ to the direction of specular reflection $\vec{\omega}_r = (\theta_{in}, \pi)$:

$$S = (R_z(\pi) \cdot R_y(\theta_{in}))^T = \begin{pmatrix} -\cos\theta_{in} & 0 & -\sin\theta_{in} \\ 0 & -1 & 0 \\ -\sin\theta_{in} & 0 & \cos\theta_{in} \end{pmatrix}. \tag{15}$$

After rotation, the spherical coordinates of the view direction $\vec{v}$ are related to $\vec{\omega}_r$, therefore they are $\vec{\omega}_d$. Hence, the normalized texture coordinates are calculated using this formula:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \frac{f_\phi(S\cdot\vec{v})}{2\pi} \\ \frac{f_\theta(S\cdot\vec{v})}{\pi} \\ \frac{f_\theta(\vec{l})}{\theta_{in,m}} \end{bmatrix}, \tag{16}$$

where the function f maps Cartesian coordinates to spherical, $\vec{l}$ and $\vec{v}$ represent the light and the view direction in the introduced tangent space.

To perform testing and performance evaluations, the same testing scene was established in both game engines. This scene contains six models (specifically Stanford Bunny [18], Stanford Armadillo [9], Phlegmatic Dragon [7], Stanford Dragon [3], Utah Teapot [13] and Spot [2]) with six different materials. The rendered image from this scene is shown in Figure 7.

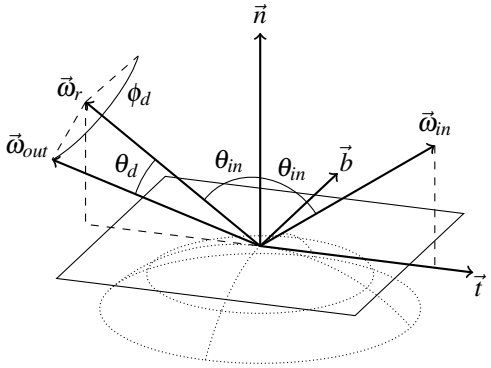


Figure 8: The tangent space used for BRDF evaluation.

5.2 Unity

Unity [20] has three rendering pipelines: build-in, universal (URP), and high definition (HDRP). The shader for rendering measured BRDF was implemented for the built-in render pipeline because modifying a BRDF evaluation in other pipelines is not officially supported and requires a bit of reverse engineering [22].

The built-in rendering pipeline uses forward rendering by default, and therefore the implementation of the shader was straightforward. Each material has an input texture with BRDF data and evaluates the lighting in the fragment shader for each light that affects the object [19].

5.3 Unreal Engine 5

Unreal Engine [6] has a pipeline based on deferred shading. A shader, editable by a user, writes the necessary data for lighting (such as position, normal, and BRDF parameters) to the G-buffer (an off-screen framebuffer) [12]. In the G-buffer, it is not feasible to store the whole texture with measured BRDF data. Instead, it is necessary to store only an index of a sample in the atlas.

(a) Render with measured BRDF materials.

(b) Render with the Lafortune-Phong materials.

Figure 7: The test scene rendered in Unity (it appears almost identical in Unreal Engine). Differences between the analytical model and measured BRDFs are mostly noticeable in a specular part, where the highlights of the Lafortune-Phong model do not have soft endings.

During the second pass, lighting is calculated for every pixel on the screen based on data stored in the G-buffer. This pass is a part of the rendering engine, and making changes to the BRDF evaluation requires directly editing the source code of the engine. It involves modifying the shader and structure of the G-buffer, editing the UI components in the material editor, and incorporating some logic for manipulation with the atlas [1].

6 Discussion

6.1 Performance

The rendering speed was evaluated in the test scene in both game engines. The shader using measured BRDF was compared with the shader using the Lafortune-Phong analytic model [10]. This choice was determined due to differences in default shader models between game engines. Additionally, default shaders provide support for indirect lighting, which the current shader implementation does not offer.

Performance measurements were performed on a Linux PC with Intel i5-9600K @ 4.5 GHz CPU and Nvidia GeForce GTX 1660 GPU. Both game engines use the Vulkan API for rendering. The measured BRDF shader is slightly slower (approx. $2-3\%$).

	Unity 2022.1.19f1	Unreal Engine 5.1.1
Measured BRDF	10.05 ms (99.5 fps)	12.37 ms (80.9 fps)
Lafortune-Phong	9.72 ms (102.9 fps)	12.12 ms (82.5 fps)

Table 1: Average rendering time of the test scene with three directional lights on 4K resolution. The average was calculated from 15 seconds run with 5 seconds warm-up.

6.2 Drawbacks

The primary disadvantage of this method is the increased memory usage caused by the requirement to store the LUT texture. Each measurement from the created dataset took around 1 MB of VRAM ($361 \cdot 181 \cdot 4$ pixels in RGB9_E5 format, 4 bytes per pixel), but a LUT texture with denser measurements (which means higher resolution of the texture) will require significantly larger amounts of memory.

Another drawback is the restriction of rendering only direct lighting. For performance reasons, game engines make some assumptions that cannot be easily fulfilled with tabularized BRDF data. For example, Lumen Global Illumination in Unreal Engine uses a single simplified analytic model [5].

Figure 9: Miniatures of all measured samples in the dataset.

6.3 Future work

One of the possible improvements is to expand the implemented shader to support anisotropic materials. It will require manually rotating the scatterometer or another scatterometer with anisotropic support and extending the LUT texture to four dimensions. Because 4D textures are not generally supported, performing linear interpolation along a single axis needs to be realized within a shader by additional steps. Furthermore, memory consumption will increase even further, necessitating the implementation of some form of data compression.

7 Conclusion

In the preceding sections, we explained the utilization of measured reflectance for real-time rendering in computer graphics. We described the process from acquiring measured BRDF data, and processing them, to computing the radiance of a pixel in a shader. A total of 216 material samples were measured, and a tool was developed to process and visualize the data. Although the method outlined may have some limitations, in specific scenarios, it may be more appropriate than a generic analytic model.

Acknowledgments

This work was supported by the Grant Agency of the Czech Technical University in Prague, No SGS22/173/OHK3/3T/13.

References

[1] 0ne3y3. New shading models and changing the GBuffer, 12 2022. https://dev.epicgames.com/community/learning/tutorials/2R5x/unreal-engine-new-shading-models-and-changing-the-gbuffer [Accessed 2024-03-04].

[2] Keenan Crane, Ulrich Pinkall, and Peter Schröder. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.

[3] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 303–312, New York, NY, USA, 1996. Association for Computing Machinery.

[4] Jonathan Dupuy and Wenzel Jakob. An adaptive parameterization for efficient material acquisition and rendering. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 37(6):274:1–274:18, November 2018.

[5] Epic Games. ShadingModels.ush, Unreal Engine source code. https://github.com/EpicGames/UnrealEngine/blob/5ca9da84c694c6eee288c30a547fcaa1a40aed9b/Engine/Shaders/Private/ShadingModels.ush#L343 [Accessed 2024-03-04].

[6] Epic Games. Unreal Engine, 1998–2024. https://www.unrealengine.com/en-US.

[7] Jiří Filip, Radek Holub, Vlastimil Havran, Jaroslav Křivánek, and Daniel Sýkora. Phlegmatic Dragon, 2007. https://web.archive.org/web/20220710054500/https://dcgi.fel.cvut.cz/cgg/eg07/index.php?page=dragon [Accessed 2022-07-10].

[8] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, page 143–150, New York, NY, USA, 1986. Association for Computing Machinery.

[9] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 313–324, New York, NY, USA, 1996. Association for Computing Machinery.

[10] Eric Lafortune and Yves Willems. Using the Modified Phong Reflectance Model for Physically Based Rendering. Technical Report CW 197, Department of Computing Science, K.U. Leuven, 11 1994.

[11] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.

[12] Michael Muir. Unreal Engine Lighting. https://dev.epicgames.com/community/learning/tutorials/Le7b/unreal-engine-lighting [Accessed 2024-03-04].

[13] Martin Newell. Utah Teapot, 1975. https://graphics.cs.utah.edu/teapot/ [Accessed 2024-03-09].

[14] F.E. Nicodemus, J.C. Richmond, J.J. Hsia, W.I. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. *Applied Optics*, 9:1474–1475, 1977.

[15] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 6 1975.

[16] Christophe Schlick. An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum*, 13(3):233–246, 1994.

[17] Synopsys. *MiniDiff V2 User's Manual*, 2021.

[18] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, page 311–318, New York, NY, USA, 1994. Association for Computing Machinery.

[19] Unity Technologies. Rendering paths in the Built-in Render Pipeline. `https://docs.unity3d.com/2022.1/Documentation/Manual/RenderingPaths.html` [Accessed 2024-03-03].

[20] Unity Technologies. Unity, 2005–2024. `https://unity.com/`.

[21] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. page 195–206, 2007.

[22] Bronson Zgeb. Custom Lighting in URP with Shader Graph, 2021. `https://bronsonzgeb.com/index.php/2021/10/04/custom-lighting-in-urp-with-shader-graph/` [Accessed 2024-03-04].