A unified GLTF/X3D extension to bring Physically-Based Rendering to the Web

Timo Sturm* Miguel Sousa† Maik Thöner‡ Max Limper§

Fraunhofer IGD, Darmstadt / TU Darmstadt

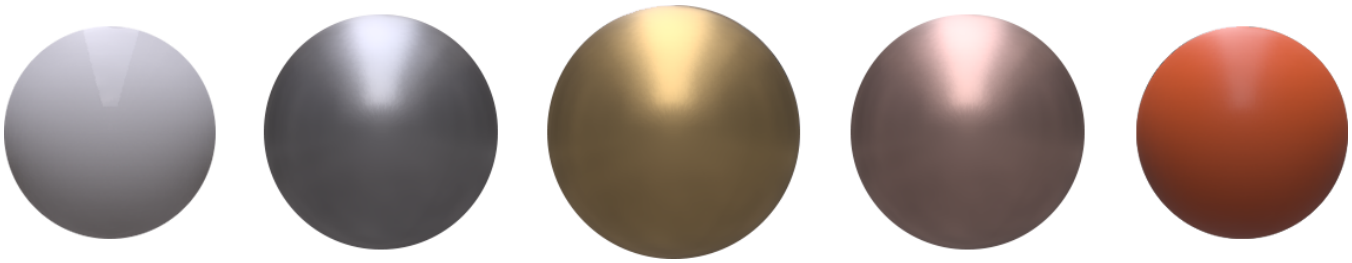

Figure 1: *Dielectric and metallic physically-based materials, described with a small set of intuitive parameters and rendered in real-time inside a Web browser, using WebGL. From left to right: white dielectric, iron, gold, copper, red dielectric.*

Abstract

We present a unified material description and transmission format for real-time, physically-based shading. Our format is general enough to be used consistently across multiple rendering systems and platforms, covering a wide range of applications from desktop to Web. Furthermore, our format's expressiveness allows to represent a wide variety of real-world materials. First, we define a common parameter set for physically-based shading in modern, real-time 3D graphics systems. We then demonstrate its applicability for several types of materials on different rendering systems. Finally, we propose a transmission format, in the form of extensions for the glTF and X3D standards.

Keywords: physically-based rendering, real-time rendering, glTF, X3D standardization

Concepts: •**Computing methodologies** → **Graphics file formats;** Reflectance modeling;

1 Introduction

Ever since the first computer generated images of 3D models, there has been an attempt to improve the shading of the surfaces to produce a more realistic result.
Based on the Phong reflection model [Phong 1975], the Blinn-Phong model [Blinn 1977] became ubiquitous in real time 3D graphics, being used by both, OpenGL and Direct3D.
Even though improved models have been introduced over the years, these were not a good fit for real-time applications as they added significant computational costs.
The introduction of programmable graphics pipelines made it possible to use variations of the reflection models better suited for a particular application. Meanwhile, the graphics APIs moved away from fixed lighting models and, as a result, no new standard models were established.
In recent years, Physically-Based Rendering (PBR) has been gaining a lot of attention. The main premise is that modeling natural light behavior leads to more consistent results and unifies the shading of diverse materials. In this context, Disney's work [Burley 2012] was highly influential.
Although the number of content creation tools and game/rendering engines implementing the concept of PBR increased, material definitions converged to a common set of parameters. Nonetheless, no proposal for a standardized material description has been put forward.
With the introduction of WebGL, 3D models became commonplace on the Web with some of the Web applications behind it using PBR. As in the case of desktop graphics, no unified model for materials has been suggested.

In this paper we propose a set of parameters and a PBR shading model that we believe can be used for representing a wide range of materials, based on the current state of the art of PBR. Our solution results from capturing the most relevant features from prominent implementations of physically-based rendering.
Furthermore, we present a set of nodes for X3D and a corresponding set of extensions for glTF for representing PBR materials, along with the resulting renderings captured in a Web context.

2 Related Work

PBR Engines. Most major digital content creation tools and game engines currently make use of some form of physically-based rendering.
Unreal Engine 4 [Epic 2016], the game engine from Epic Games, uses PBR extensively. The use of PBR led to more realistic rendering while at the same time it improved artists workflow by simplifying the process of creating materials. These materials are defined by a small set of intuitive parameters.
Unity 5 [Unity 2016], another popular game engine, also uses PBR and a small set of parameters in its Standard Shader in order to produce realistic rendering. The move to PBR also simplified the process of authoring physically plausible materials compared with previous versions of the engine.

*e-mail:timo.sturm@igd.fraunhofer.de
†e-mail:miguel.de.sousa@igd.fraunhofer.de
‡e-mail:maik.thoener@igd.fraunhofer.de
§e-mail:max.limper@igd.fraunhofer.de

Web3D '16, July 22-24, 2016, Anaheim , CA, USA
 ISBN 978-1-4503-4428-9/16/07 . . . $15.00
DOI: http://dx.doi.org/10.1145/2945292.2945293

Substance Designer 5 [Allegorithmic 2016] is a material authoring tool designed around PBR. Materials are created using a node editor and the application can generate a wide range of textures maps. It integrates with the most popular digital content creation tools and game engines.

WebGL engines. Currently, most engines running on WebGL are not based on PBR. The popular three.js engine [ThreeJS 2016] does not have native support for physically-based shading although it is possible to implement it by writing custom shaders and materials. The X3DOM framework [Behr et al. 2009] is a similar example. Although it has an implementation of the CommonSurfaceShader [Schwenk et al. 2012], it is not complete thus far and it only supports some of the general parameters from the CommonSurfaceShader node.
However, some WebGL based applications are introducing PBR on the Web. One example is Sketchfab [Sketchfab 2016], a website for displaying and sharing 3D content. It allows the use of PBR materials on the uploaded models.
Marmoset, developer of Toolbag [Marmoset 2016], also uses PBR in their pipeline. Marmoset Viewer is a WebGL based viewer using PBR for content exported by Toolbag 2.

Transmission Formats. The GL Transmission Format (glTF) [Khronos a] is an open project by the Khronos Group which aims at defining a format for asset delivery tailored for OpenGL APIs. One key characteristic of the glTF is that its data structures are homologous to the ones used by OpenGL.
In the context of Web applications, often running with limited resources, this is an important feature as it significantly simplifies the parsing of data structures and eliminates the need for data reencoding, especially when using its binary version as defined in the `KHR_binary_glTF` extension.
Another important aspect of this format is its extensibility. The glTF extension model [Khronos c] allows the expansion of any part of the original format.
The current version of glTF (1.0) does not permit the generic definition of materials using a set of parameters although the proposed `KHR_materials_common` extension provides such support for legacy reflection models such as the Blinn-Phong model.
X3D, a standard XML-based file format for 3D content, has no support for PBR materials. However, being highly extensible, it is possible to write new Nodes to this end.

3 Physically-based Rendering

Physically-based rendering (PBR) refers to the concept of using realistic shading/lighting models along with measured surface values to accurately represent real-world materials.

PBR is more of a concept than a strict set of rules, and as such, the exact implementations of PBR systems tend to vary. Still, as they are based on the same principal idea (improve realism by approximating physical laws), they are similar across implementations. Some of the main goals behind PBR are:

Simplicity

PBR uses an easy to understand material description defined by a small set of intuitive parameters instead of a large array of parameters, which results in decision paralysis, trial and error, or interconnected properties that require many values to be changed for a single intended effect.

Extensiveness

PBR can cover up most of the materials that occur in the real world with a single shading model. As deferred shading limits the number of shading models that can be used, this is highly beneficial. On forward renderers it improves performance by reducing shader switching.

Consistency

By using physically-based shading models, which follow real physical laws, materials will look accurate and consistent in all lighting conditions without changing an immense list of parameters and settings.

4 Shading/Lighting Model

The core of every PBR implementation is the underlying shading model. As mentioned before, at this time PBR is more of a concept than a true standard. Therefore, the implementation of the shading/lighting model and the used diffuse and specular Bidirectional Reflectance Distribution Function (BRDF) varies between systems.

Most systems we analyzed implement PBR as a combination of Imaged-based lighting and the the Cook-Torrance microfacet specular BRDF [Cook and Torrance 1982] and the Lambertian diffuse BRDF.

4.1 Specular BRDF

The Cook-Torrance reflectance model was chosen as it is the most commonly used physically-based specular BRDF. It is based on the microfacet theory in which surfaces are composed of small-scale planar detail surfaces of varying orientation. Each of these small planes, so called microfacets, reflects light in a single direction based on its normal.

The Cook-Torrance specular BRDF is defined as follows:

$$f(l,v) = \frac{D(h)F(v,h)G(l,v,h)}{4(n\dot{l})(n\dot{v})} \tag{1}$$

Where l is the light direction, v is the view direction, h is the half-vector, n is the normal, D is the normal distribution function (NDF). F is the Fresnel term and G is the geometry term.

Specular D

Specular D is represented by the normal distribution function which is used to describe the statistical orientation of the micro facets at a given point. The first PBR implementations used distributions such as Phong or Beckmann, but recently the GGX distribution [Walter et al. 2007] has become a popular choice. It is defined by:

$$D(h) = \frac{\alpha^2}{\pi\left((n\cdot h)^2(\alpha-1)+1\right)^2} \tag{2}$$

Where h is the half-vector(microfacet normal), n is the normal and α is the roughness of the material.

Specular F

The specular F term represents the Fresnel function. The Fresnel function is used to simulate the way light interacts with a surface at different viewing angles. We adopt Schlicks Approximation [Schlick 1994] for the Fresnel term which is the most commonly used in 3D graphics.

$$F(v,h) = F_0 + (1-F_0) * (1-v\cdot h)^5 \tag{3}$$

Where F_0 is the specular reflectance at normal incidence.

Specular G

Specular G represents the geometry shadowing function used to describe the attenuation of the light due to microfacets shadowing each other. This is once again a statistical approximation which models the probability of energy loss. This may occur due to microfacets being occluded by each other or light bouncing between multiple microfacets, before reaching the observer's eye. The geometry attenuation is derived from the normal distribution function. Most implementations use Smith's shadowing function [Walter et al. 2007] or Schlick's model [Schlick 1994].

The complete geometry shadowing function is composed of the two partial functions $G_1(n,l)$ and $G_1(n,v)$ as follows:

$$G(l,v,h) = G_1(n,l)G_1(n,v) \tag{4}$$

The partial Smith shadowing function is defined as:

$$G_1(n,v) = \frac{2(n\cdot v)}{(n\cdot v)\sqrt{\alpha^2+(1-\alpha^2)(n\cdot v)^2}} \tag{5}$$

and the partial Schlick shadowing function is defined by:

$$k = \frac{(\alpha+1)^2}{8} \tag{6}$$

$$G_1(n,v) = \frac{(n\cdot v)}{(n\cdot v)(1-k)+k} \tag{7}$$

4.2 Diffuse BRDF

The Lambertian diffuse BRDF is still the first choice. Even though other models (e.g. [Burley 2012]) are more accurate, the visual improvements are arguably insufficient for justifying the extra computation in real-time applications.

The Lambertian diffuse is defined as:

$$f(l,v) = \frac{c_{diff}}{\pi} \tag{8}$$

Where c_{diff} is the diffuse reflected color of the material. In order to ensure energy conservation, the diffuse term should be balanced using the inverse of the Fresnel term from the specular component [Shirley et al. 1997]:

$$f(l,v) = (1-F(v\cdot h))\frac{c_{diff}}{\pi} \tag{9}$$

4.3 Imaged-based Lighting

Image-based lighting (IBL) is the most common technique to simulate indirect lighting in the current PBR engines. It uses environment maps from real-world light probes or rendered scenes to illuminate objects.

Importance Sampling

To use the presented shading model with imaged-based lighting, the radiance integral needs to be solved, which can be achieved by using importance sampling. Importance sampling substantially improves the Monte Carlo algorithm by introducing a guided approach to the sampling. The idea is that we can define a Probability Distribution Function (PDF) that describes where we want to sample more and where we want to sample less.

The following equation describes the numerical integration:

$$\int_H L_i(l)f(l,v)\cos\theta_l dl \approx \frac{1}{N}\sum_{k=1}^{N}\frac{L_i(l_k)f(l_k,v)\cos\theta_{l_k}}{p(l_k,v)} \tag{10}$$

which can be solved in real-time directly on the GPU [Colbert and Kivnek 2008].

But even with importance sampling, many samples are still needed to produce acceptable results. In simple scenes with only a few objects and a single environment map this is not a problem. But in more complex scenes with many different objects and multiple environmental light sources the pure importance sampling approach is not suitable anymore for real-time rendering.

This problem can be solved using a split sum approximation [Karis 2013]. This new technique is employed in the Unreal Engine 4 for real-time PBR of complex scenes.

Split Sum Approximation

The split sum approximation splits the sum from (10) into a product of two sums, both of which can be pre-calculated, see (11). This approximation is exact for a constant $L_i(l)$ and fairly accurate for common environments.

$$\left(\frac{1}{N}\sum_{k=1}^{N}L_i(l_k)\right)\left(\frac{1}{N}\sum_{k=1}^{N}\frac{f(l_k,v)\cos\theta_{l_k}}{p(l_k,v)}\right) \tag{11}$$

The first sum is pre-calculated for different roughness values by convolving the environment map with the GGX distribution using importance sampling and storing the results in individual mipmap levels of an environment map texture, as shown in figure 2.

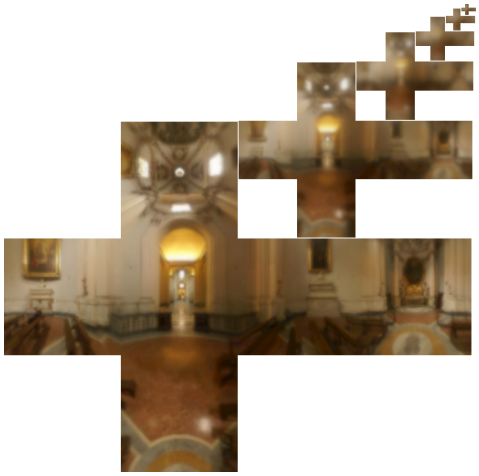

Figure 2: *Pre-calculated environment map, with varying roughness levels stored in different mipmap levels.*

The second sum in (11) includes the remainder and is equivalent to integrating the specular BRDF with a solid-white environment. By substituting in Schlicks Fresnel approximation (3) into the left hand side of (10) F_0 can be factored out of the integral. This leaves two

inputs (roughness and $\cos\theta_v$) and two outputs (a scale and bias to F_0), which can also be pre-calculated and stored in a 2D Look-Up Texture (LUT), as shown in figure 3.

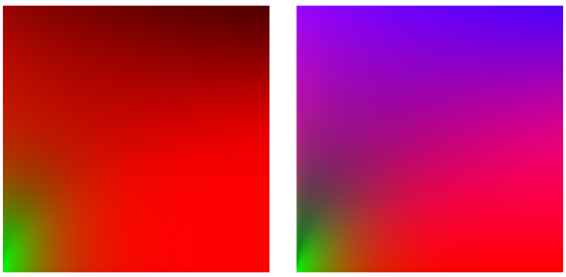

Figure 3: *LUT of the second sum. Left Schlick and right Smith.*

The main advantage of the pre-calculated LUT is that it is constant for white light and it does not depend on a specific environment. So it has to be pre-calculated only once for a particular shading model and can be reused in every shader.

Therefore, with the split sum approximation, only two texture fetches per fragment are needed to calculate the specular color. This is a significant improvement over the importance sample method, which requires multiple samples per pixel.

5 Material Model

In order to transport material specific data that can be fed into the rendering pipeline, a material model is needed. We already mentioned in Section 3 that PBR is a methodology and, although principles and guidelines exist, there is no true standard. Therefore, there is also no single standardized material model for PBR at this time. Nevertheless, two models became commonplace [Allegorithmic 2015], namely the **specular-glossiness** and the **metal-roughness** model. Both models represent the same data, but they transport it in different ways.

5.1 Specular - Glossiness

The base of the specular-glossiness material model consists of the following three parameters:

- Diffuse
- Specular
- Glossiness

In the specular-glossiness model the diffuse value represents, similar to its traditional counterpart, the reflected diffuse color of the material expected for pure black (0.0). In the specular-glossiness model pure black (0.0) indicates raw metal, since metal doesn't have a diffuse component. In some implementations the diffuse value is also called albedo.

The specular value defines the F_0 value for dielectrics and the reflectance values for metals. The F_0 for dielectrics will be a grayscale value and the metal reflectance can be colored as some metals absorb light at different wavelengths.

The Glossiness value is a factor in $[0, 1]$ and describes the surface irregularities that cause light diffusion. Rougher surfaces will have wider and dimmer looking highlights. Smoother surfaces will keep specular reflections focused. A value of 0.0 (black) represents a rough surface and a value of 1.0 (white) represents a smooth surface.

Figure 4 shows the three single components of the metal-roughness model and the rendered result.

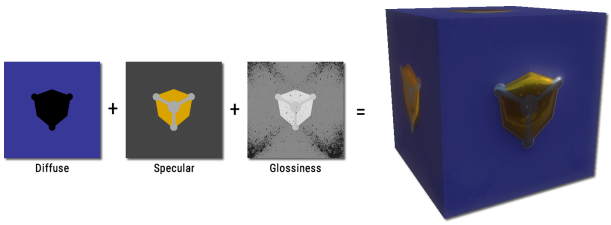

Figure 4: *Illustration of the Specular-Glossiness model.*

5.2 Metal - Roughness

The base of the metal-roughness material model consists of the following three parameters:

- BaseColor
- Metallic
- Roughness

In the metal-roughness model the base color is used to transport two different forms of data depending on the material type. For metallic materials the base color transports the specific measured reflectance value F_0, for dielectrics it transports the reflected diffuse color of the material. In this model it is not possible to specify a reflectance value for non-metals. For those a constant reflectance value of 4% (0.04) is usually used and covers most common dielectric materials.

The Metallic value is a factor in $[0, 1]$ and it acts similarly to a mask which tells the shader how it should interpret data found in the basecolor. While it is possible to specify values between 0 and 1, it is often used as a binary parameter with a value of either 0 or 1.

The Roughness value in the metal-roughness model is simply the inverted glossiness value in the specular-glossiness and also describes the surface irregularities that cause light diffusion.

Figure 5 shows the three single components of the metal-roughness model and the rendered result.

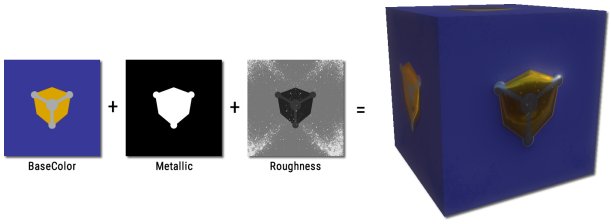

Figure 5: *Illustration of the Metal-Roughness model.*

5.3 Reflectance Values

Since PBR is based on physical laws one cannot use arbitrary inputs for the reflectance values. Especially for the specular-glossiness model where the parameters allow full control over the reflectance of both metals and non-metals. The values must be correct and measured from real world data. Fortunately, several reference charts

exist such as the one from [Dontnod 2014] which provides sets of values for specific materials.

5.4 Material Model Comparison

Both models presented have advantages and disadvantages. Nonetheless, we consider the metal-roughness model more appropriate for a new PBR material standard. In our opinion, the only significant advantage of the specular-glossiness model is the full control of the F_0 value for dielectric materials. But this is also a drawback since it does not guarantee energy conservation for all values. For example, a white (1.0) diffuse and a white (1.0) specular value can be combined reflecting/refracting more light than was initially received if the underlying implementation (the shader in particular) does not compensate for these situations. In contrast the constant F_0 value of the metal-roughness model is less error prone and it is nonetheless possible to cover the most common dielectric materials with a F_0 value of 4%.

In addition, the parameter names of the specular-glossiness can be more confusing as it uses a terminology similar to traditional models like Blinn-Phong but requires different data, whereas the parameter names of the metal-roughness model are easy to understand and very clear in their meaning. Moreover, the metal-roughness model is more memory friendly, as metallic and roughness are both single floating point values.

Overall the difference in the two models is narrow and applies only to certain edge cases. Therefore, in favor of simplicity we propose to use the metal-roughness model as described in the next section.

6 Transmission Formats

To enable the widespread use of the model in different applications, a well-defined and accessible transmission format is required. In the web glTF and X3D have emerged as two of the most used 3D transmission formats. Fortunately they are also easy to adapt, so we propose extensions for these two formats in order to transmit the required information for the shading and materials of the models.

As described in Section 5 we only need a very limited set of parameters for the material description and three textures to define the environmental light information in the scene.

For every material description we need parameters for albedo, metallic and roughness. All three values can be passed in as a single field value that is then used for the full mesh. Alternatively, a texture can be used to define a non uniform material on a mesh. The environmental light textures are stored as light information in the formats.

With this very small, well-defined and representation independent set of parameters it is simple to also add it in any custom transmission format and to convert it from one format to another, enabling a even wider use in different already existing applications.

6.1 glTF

glTF 1.0 doesn't specify a shading model for materials but relies instead on GLSL shader code and parameters (uniforms) in order to describe a particular appearance. The drawback of this approach is that the shader code is highly dependent on the renderer (e.g. forward renderer vs. deferred renderer) and thus not compatible across rendering engines.
The `KHR_materials_common` extension [Khronos b] provides a straightforward alternative for defining materials based on the popular Blinn, Phong and Lambert shading models as well as solid colors (no shading). Each of these shading models only require a compact set of parameters. An advantage of such a high level definition is that it is implementation independent.

We drafted a glTF extension, written against glTF version 1.0, which expands the glTF material schema by adding a new identifier for the technique property ("PBR") and specifying our model's properties (color, metallic, and roughness) within the values property used in our material description.

Since Image Based Lighting plays an important role in generating plausible rendering results, we further outlined another extension for defining prefiltered environment diffuse and specular textures. The current glTF specification only defines 2D textures but there are plans to support cube map textures in the future.

Directional, point or spot light sources should also be supported by a renderer implementing this extension and can be defined using the `KHR_materials_common` extension.

The following glTF excerpt contains the relevant elements for defining a physically based material using our extension, including the prefiltered environment maps.

```
{
  "extensionsUsed": [
    "YYZ_materials_pbr",
      "YYZ_lights"
  ],
  "materials": {
    "pbr_plastic" : {
      "extensions": {
        "YYZ_materials_pbr": {
          "technique": "PBR",
          "values": {
            "albedo": "plastic_albedo_tex",
            "metallic": 0.0,
            "roughness": 0.5
          }
        }
      }
    }
  },
  "extensions": {
    "YYZ_image_based_lighting": {
      "lights": {
        "env_light": {
          "type": "ibl_pbr",
          "diffuse": "env_diffuse_tex",
          "specular": "env_specular_tex",
          "brdf": "env_brdf_text"
        }
      }
    }
  },
  "scene": "default_scene",
  "scenes": {
    "default_scene": {
      "ibl": [
        "env_light"
      ]
    }
  }
}
```

The values for color, metallic, and roughness can be scalars (a 4 component vector in the case of color) or textures. If textures are used, the mesh should provide the necessary texture coordinates. It is assumed that all textures share the same texture coordinates, defined as the primitives attribute `TEXCOORD_0`.

6.2 X3D

For X3D we propose two new nodes to represent the material and shading properties for the PBR materials.

The first node is the PhysicalMaterial node that is used to specify the parameters for a surface. It is used as alternative to the Material node. The attributes are albedoFactor, roughnessFactor and metallicFactor that represent the albedo, roughness and metallic values as desribed in Section 5. All three parameters can optionally be replaced by a texture representation with the corresponding names albedoMap, roughnessMap and metallicMap. The textures are provided by the ImageTexture node. If any parameter is specified with a texture, texture coordindates for the vertices are required.

The second node we propose is the PhysicalEnvironmentLight node. The PhysicalEnvironmentLight node is used to specify the environmental textures for the shading and is placed inside a scene node. The attributes are diffuse, specular and brdf LUT. The diffuse and specular fields contain the baked lighting information in form of a ImageCubeMapTexture and the brdf slot the lookup table for the brdf function as described in Section 4.

X3D-based PhysicalMaterial with only factors

```
<PhysicalMaterial albedoFactor="0.22 0.3 0.5"
             roughnessFactor="0.5",
             metallicFactor="1.0" />
```

X3D-based PBRMaterial with textures

```
<PhysicalMaterial>
    <ImageTexture
    url="albedo.png"
    containerField="albedoMap" />
    <ImageTexture
    url="roughness.png"
    containerField="roughnessMap" />
    <ImageTexture
    url="metallic.png"
    containerField="metallicMap" />
</PhysicalMaterial>
```

X3D-based PhysicalEnvironmentLight for IBL

```
<PhysicalEnvironmentLight>
<ImageCubeMapTexture
   url="diffuse_env.dds"
   containerField="diffuse" />

<ImageCubeMapTexture
   url="specular_env.dds"
   containerField="specular" />

<ImageTexture
   url="brdf.png"
   containerField="brdf" />
</PhysicalEnvironmentLight>
```

7 Implementation

After our analysis of the current state of the art on PBR and respective implementations, we were able to bring it to the Web. GPU-based tools for pre-calculating the environment map textures and look-up textures were developed. Our glTF and X3D loaders were extended to support the new material definitions and our rendering engine was updated with new shaders.

We used WebGL 2 as a base for our pipeline. Although it would be possible to use WebGL 1.0 while keeping all the functionality, features available in WebGL 2, such as multiple render targets and floating point texture support, allow for a straightforward implementation.

The initial results were very promising. From our tests, the performance of current WebGL implementations, running on a common desktop computer, is appropriate for PBR rendering. The visual improvements can be observed in Figure 6, where our PBR rendering pipeline is compared with a traditional non-PBR pipeline (x3dom). The 3D model used in the comparison is the popular and publicly available PBR ready model by [Maximov 2014] which is also shown in Figure 10 under different lighting conditions.

Using our implementation we tested several dielectric and metallic materials (Figure 1 and Figure 7) with varying roughness levels (Figure 8) and under different lighting conditions (Figure 9).

The current implementation is under active development and there are still many aspects to be improved. Analytical direct lighting is one of the open topics as well as translucent materials. We hope that the approach discussed in this paper will help bringing more physically-based rendering systems to the Web.

Figure 6: *Comparison of a traditional non-PBR pipeline (top) with our new state of the art PBR pipeline (bottom).*

8 Conclusion & Future Work

In this paper, we presented a common material description for physically-based rendering on the Web and proposed extensions for the glTF and X3D Web transmission formats. The material description we settled upon is based on the parameter sets used in most modern physically-based rendering engines. The introduction of a unified material representation for the transmission formats ensures consistent results across rendering systems. For glTF, our proposed extension enables the transport of a wide array of materials without the need for custom shaders. This notably reduces the size and

complexity of the transferred data for all supported materials, while removing the constraints of the default glTF shader based material definition, which is inherently renderer dependent. In the case of X3D, we represent these materials with a set of new nodes.

Although the presented model allows to reproduce a wide variety of real-world materials, we would like to expand it in the future. Adding layering techniques would further increase the set of representable materials. This would allow to model surfaces with multiple layers that interact with the incoming and outgoing light.

Another important topic which is related to PBR is direct lighting. Conventional direct light sources like directional, point, or spot lights are generally not suitable for PBR, since they can quickly break the law of energy conversation. As such, analytical area lights suitable for use in Web platforms would be another topic for further research. Finally, we would like to investigate the conversion from different sources of material representations.

References

ALLEGORITHMIC, 2015. The comprehensive pbr guide vol. 2. https://www.allegorithmic.com/pbr-guide.

ALLEGORITHMIC, 2016. Substance designer. http://www.allegorithmic.com/products/substance-designer.

BEHR, J., ESCHLER, P., JUNG, Y., AND ZÖLLNER, M. 2009. X3dom: A dom-based html5/x3d integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, ACM, New York, NY, USA, Web3D '09, 127–135.

BLINN, J. F. 1977. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '77, 192–198.

BURLEY, B. 2012. Physically-based shading at disney, part of practical physically based shading in film and game production. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12.

COLBERT, M., AND KIVNEK, J. 2008. Gpu-based importance sampling. In *GPU Gems 3*, H. Nguyen, Ed. Addison-Wesley, 459–475.

COOK, R. L., AND TORRANCE, K. E. 1982. A reflectance model for computer graphics. *ACM Trans. Graph. 1*, 1 (Jan.), 7–24.

DONTNOD, 2014. Dontnod. https://seblagarde.files.wordpress.com/2014/04/dontnodgraphicchartforunrealengine4.png.

EPIC, 2016. Unreal engine 4. https://www.unrealengine.com/.

KARIS, B. 2013. Real shading in unreal engine 4. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13.

KHRONOS. gltf. https://www.khronos.org/gltf.

KHRONOS. gltf common materials. https://github.com/KhronosGroup/glTF/tree/master/extensions/Khronos/KHR_materials_common.

KHRONOS. gltf extensions. https://github.com/KhronosGroup/glTF/tree/master/extensions. https://github.com/KhronosGroup/glTF/tree/master/extensions.

LIMPER, M., THÖNER, M., BEHR, J., AND FELLNER, D. W. 2014. Src - a streamable format for generalized web-based 3d data transmission. In *Proceedings of the 19th International ACM Conference on 3D Web Technologies*, ACM, New York, NY, USA, Web3D '14, 35–43.

MARMOSET, 2016. Marmoset toolbag. http://www.marmoset.co/toolbag.

MAXIMOV, A., 2014. Pbr ready game asset cerberus. http://artisaverb.info/Cerberus.html.

PHONG, B. T. 1975. Illumination for computer generated pictures. *Commun. ACM 18*, 6 (June), 311–317.

SCHLICK, C. 1994. An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum 13*, 233–246.

SCHWENK, K., JUNG, Y., VOSS, G., STURM, T., AND BEHR, J. 2012. Commonsurfaceshader revisited: Improvements and experiences. In *Proceedings of the 17th International Conference on 3D Web Technology*, ACM, New York, NY, USA, Web3D '12, 93–96.

SHIRLEY, P., SMITS, B. E., HU, H. H., AND LAFORTUNE, E. P. 1997. A practitioners' assessment of light reflection models. In *5th Pacific Conference on Computer Graphics and Applications (PG '97)*, IEEE Computer Society, 40.

SKETCHFAB, 2016. Sketchfab. https://sketchfab.com/.

THREEJS, 2016. Threejs. http://threejs.org/.

UNITY, 2016. Unity game engine. https://unity3d.com/.

WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, EGSR'07, 195–206.

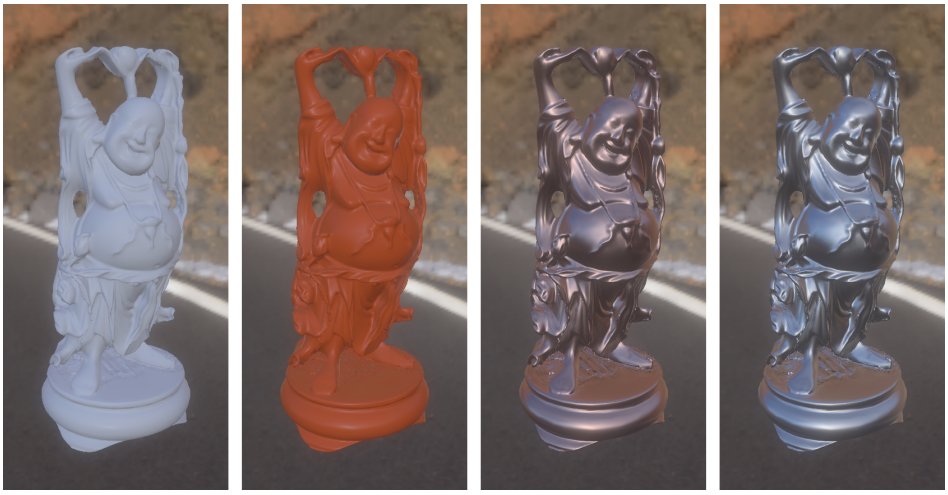

Figure 7: *Different types of Materials. From left to right: white and red dielectrics, copper, and silver.*

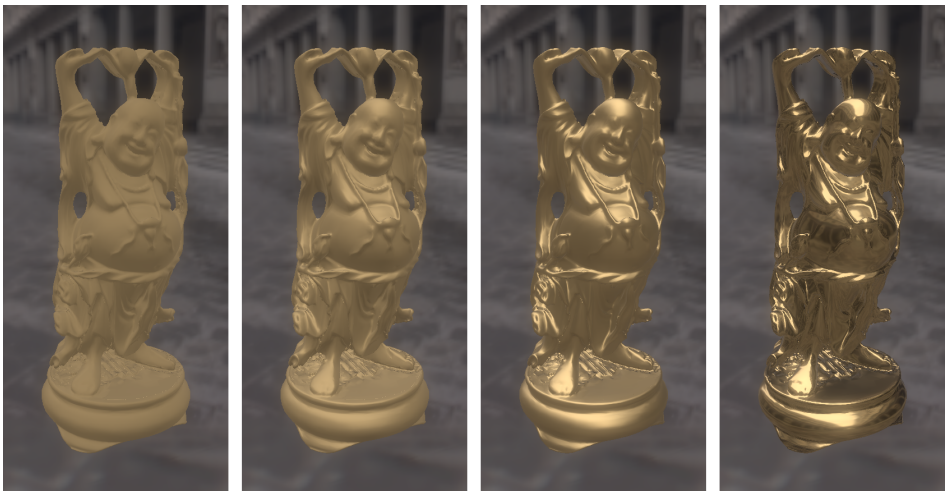

Figure 8: *Varying material roughness. From left to right: 0.7, 0.6, 0.4, 0.2*

Figure 9: *Appearance of a same material under different light conditions.*

Figure 10: *The popular Cerberus PBR ready model by Andrew Maximov rendered under two different lighting conditions.*