# Project in Visual Computing

Michael Oppitz, 1227129

October 1, 2017

**Abstract**

During the internship at ShapeDiver GmbH [1] the visual quality of an existing WebGL [2] platform had to be improved. This platform is used as an online 3D configurator, in which customers can modify the properties of parametric models. This includes the modification of properties like shape, size and materials. The difficulty for this project was to contemplate the fact that the product can be customized by the user in real-time.

## 1 Contract Requirements

The contract requirements that were specified for this project were separated into three different categories:

1. Methods for the improvement of the overall quality of the renderer. Specifically, improvements for illumination, shadows and reflections.
2. Creation of 5 optimized materials. (Wood, Gold, Fabric, Steel and Plastic)
3. The application of points 1. and 2. in 3 scenes that must be defined by the employer.

## 2 Introduction

The basis for this project was an already implemented WebGL platform, that could load and update parametric models. The goal was to get the best possible quality for the models. After some internal discussions, the choice was made to implement a standard renderer and a separate beauty renderer. The standard renderer would only be active, when a model was loading or moving. The beauty renderer would be used as soon as the objects stands still. For the beauty renderer, a small delay is acceptable to create better results.

It was necessary to create two renderers, because the quality that was needed could not be achieved in real-time on all devices. Because the models can change arbitrary while being used, the possibility to use precomputed maps like an ambient occlusion map or a light map had to be discarded.

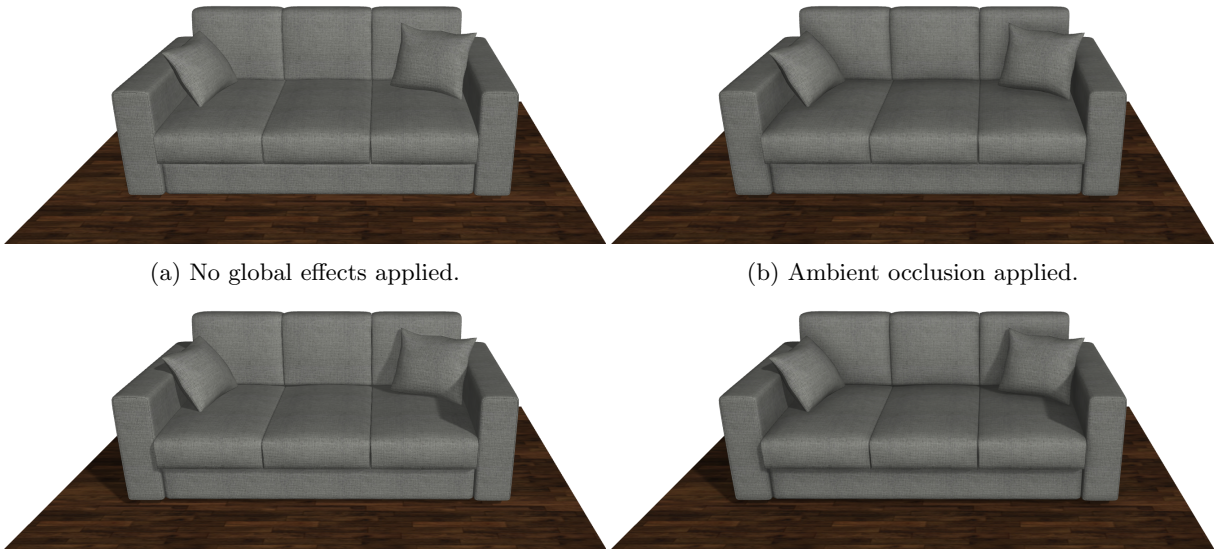
The existing framework had elements of pure WebGL functionality, but also implemented core features of the three.js library [3]. The code is written in Javascript and only small parts, that were created during this project, are written in GLSL. Most of the time the framework of the three.js library was used to implement the required functionalities to stay consistent with the pre-existing implementation.

## 3 Global Implementations

The contract requirements were separated into global implementations for shadows and ambient occlusion and a complete overhaul of the material implementation. The global implementations can be turned on and off depending on the current settings. A comparison of the different settings when switched on and off can be seen in Figure 1.

### 3.1 Shadows

The goal was to have realistic looking shadows, that can be turned on and off easily. An implementation of a basic shadow map [4] was already available in the three.js library, although it only created hard shadows. This shadow map was necessary to implement the Percentage-Closer Soft Shadows (PCSS)



(a) No global effects applied.

(b) Ambient occlusion applied.

(c) Soft shadows applied.

(d) Ambient occlusion and soft shadows applied.

Figure 1: Comparison of the different settings when switched on and off.

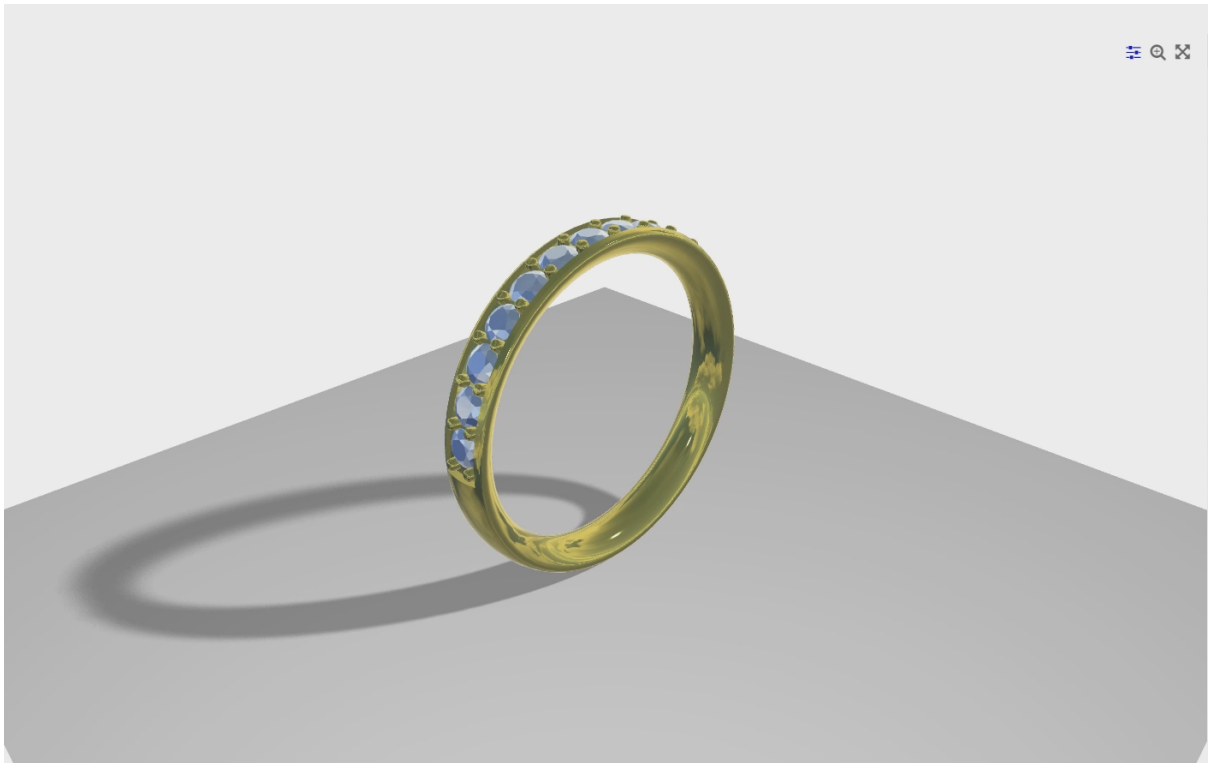


Figure 2: Soft shadows applied.

Figure 3: Ambient occlusion applied.

method [5]. The number of samples can be change interactively even while the application is launched. Although it is advised to not set the number of samples too low to avoid artefacts. Examples of this method can be seen in Figure 1c, Figure 1d and Figure 2.

To make this implementation usable in the three.js framework, a part of the shadow map implementation had to be overwritten. With a user defined value, the shadows can be turned on and off, but also the soft shadows can be activated and deactivated at runtime.

### 3.2 Ambient Occlusion

To make it possible to have occluded regions in lesser illuminated areas, ambient occlusion had to be implemented into the framework. After some research on different approaches Scalable Ambient Obscurance (SAO) [6] was chosen. This approach operates in screen-space and is therefore ideal for a scene that changes often. This approach was implemented as a pixel shader that is calculated after the scene was rendered. The results are than multiplied with the previously rendered scene.

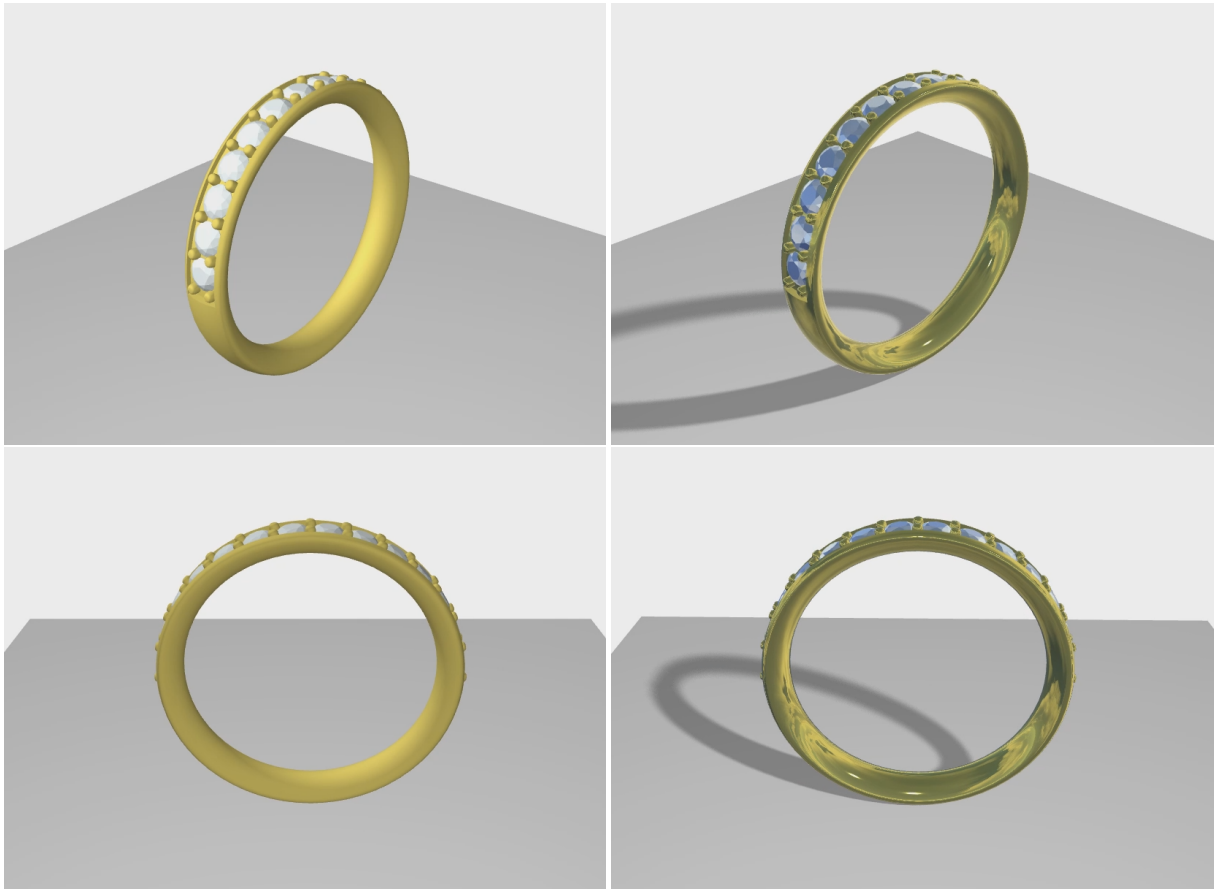
While the results could already be used in the beauty renderer, this method is still too slow for the standard renderer, because it needs a very high number of samples to converge. Even for the beauty renderer the performance was borderline. Therefore, two blur approaches were implemented. For the standard renderer, a simple gaussian blur was implemented, for which the kernel size is variable to make adjustments easier. Although the results are already very convincing, for the beauty renderer, an additional depth-dependent gaussian blur was implemented. This was necessary to get rid of ambient occlusion effects that were blurred over corners or edges, even though they were on a very different depth. Both blur filters were implemented as 1D-gaussian blurs to make use of the fact that a 2D-gaussian blur is separable into two 1D-filter, one that gets applied horizontally and one that gets applied vertically. Examples of this method can be seen in Figure 1b, Figure 1d and Figure 3.

The completed ambient occlusion process was separated into an additional render pass. Some of the implementations of the three.js libraries could be used, but most advanced implementations were insufficient in this case. Note that, at the time of the implementation only the Screen-Space Ambient Occlusion (SSAO) method [7] was available, which didn't work as it should and which produced unacceptable results.

## 4 Materials

### 4.1 Shader Change

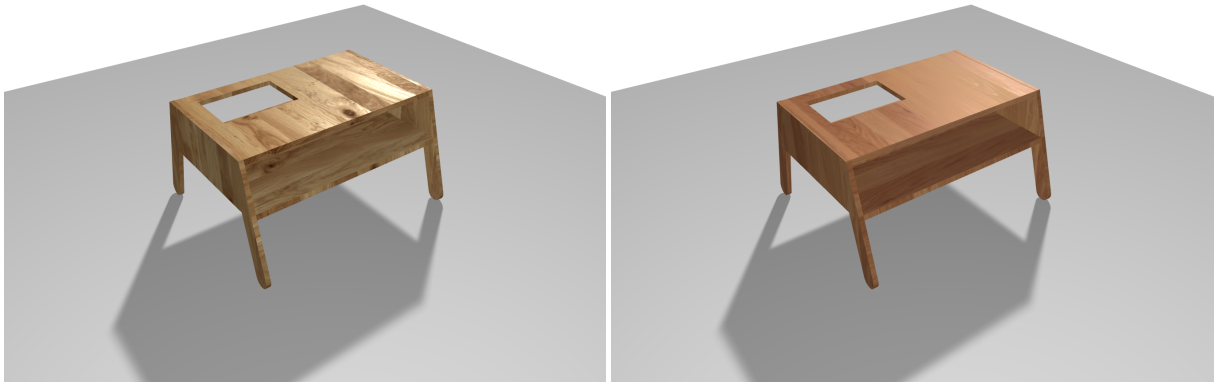
In the pre-existing framework all materials were exclusively implemented with a Blinn-Phong material shader. It was an easy decision to switch to a different material shader, because the results from the Blinn-Phong model did not produce the realistic looking results, that were needed for this project. The three.js library has already implemented a shading model that was used at Disney [8] and in the



(a) Blinn-Phong material.

(b) Physically Based material.

Figure 4: Comparison of the old and new materials.



(a) Natural oak wood material. (b) Premium oak wood material.

Figure 5: The two different wood materials.

Unreal Engine 4 [9]. The described BRDF from Disney was simplified for the Unreal Engine and was implemented in a very similar fashion in the three.js library. It is not strictly physically correct, so that artists, or in this case customers, have a certain degree of freedom in their customizations. The simplified BRDF from three.js accepts a colour, a colour map, a metalness value, a metalness map, a roughness value and a roughness map as parameters. Additionally, to these special parameters, more general additions like a normal map or an alpha map can be added.

The methods that were combined for this BRDF were:

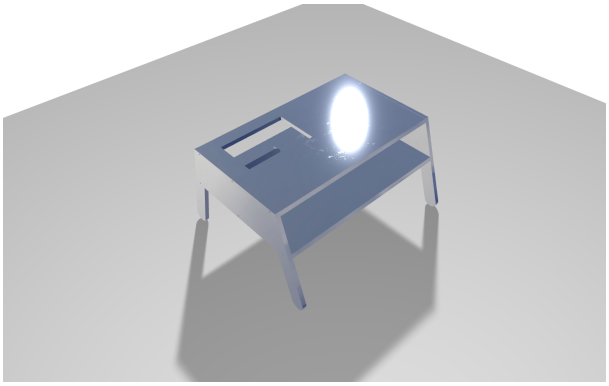
- **Diffuse BRDF:** A simple Lambertian diffuse model [10] was used in comparison to the Disney implementation where a novel empirical model was introduced.
- **Specular D:** For this term the same choice was made at Disney, the Unreal Engine 4 and three.js. The implementation of a GGX [11] / Trowbridge-Reitz [12] model was just marginally more expensive than the Blinn-Phong model, but with far more realistic looking results.
- **Specular G:** In this case a slightly altered Schlick model [13] was used. The modifications were made to better fit the Smith model for GGX [14].
- **Specular F:** The well-known Schlick's approximation [13] was used for this term. This has the advantage of not having to calculate the full Frensnel equation.

With the changes in material shader, the problem of reflections was solved, too, as in that environment maps could now be implemented for reflections. These improvements were vital to the overall appearance of the models and add another layer of realism.

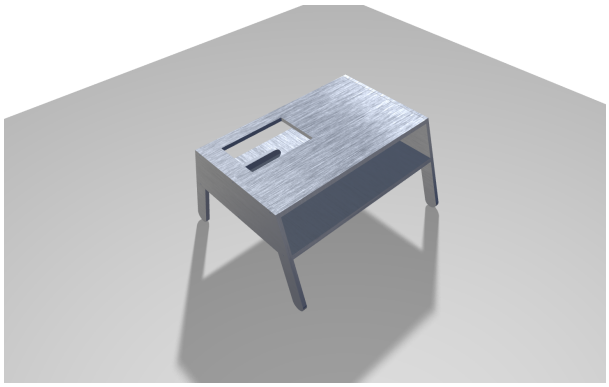
The difficulty was now to adapt these changes into pre-existing framework. The previous state was, that either a preset number or a concrete material description was attached to each model. The preset number was just an id for a preset material. These materials had to be redefined with the new PBR shaders. The case in which a material description was received had to be completely rewritten. The description still contains information that is intended for the Blinn-Phong shader, so a conversion process had to be implemented to achieve visually similar results with the new PBR shader.

## 4.2 Preset Materials

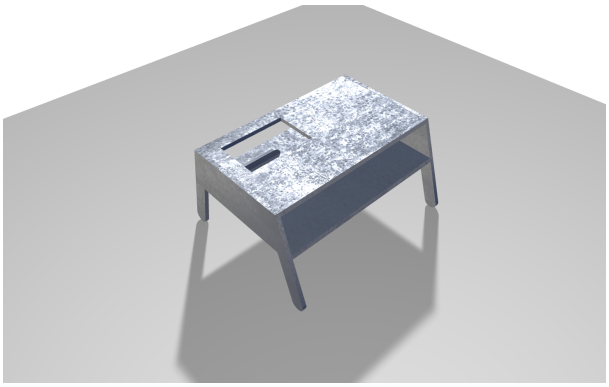
In the contract requirements, 5 different materials were mentioned to be implemented with the improved shading. Not only 5, but many more materials were implemented. All these materials are currently stored in a newly created database and functionalities were implemented to only load these materials on demand. They now serve as presets and allow the user to choose from a much more advanced selection. In Figure 5 a natural oak and a premium oak wood material can be seen and in Figure 3 the wooden floor is visible. The metal material are displayed in Figure 6, where a nearly unused, a polished, and a highly used metal can be seen. The gold material in Figure 2 is the exact same metal material as in Figure 6a only with a different colour. Additionally, there are also materials for fabrics, leather, plastic and various default materials.



(a) Nearly unused metal.



(b) Polished metal.



(c) Highly used metal.

Figure 6: The three different metal materials.

## 5 Scenes

The contract requirements restricted the work to three scenes, that were defined by the employer. In practice, it was better to implement the changes into the complete framework and not just into predefined scenes. This was done to avoid implementations that wouldn't work in other scenes and therefore wouldn't be of use to the employer. Instead, now all the previously mentioned implementations already work in a nearly all scenes. Only in very special cases some of the implementations may produce artefacts.

## 6 Future Work

Already after the first results were visible, a discussion was held about further continuation of the project after the end of the actual deadline. The decision makers of ShapeDiver were all astonished by the great improvements, that a few of them didn't even think were possible to this extent. Therefore, the verdict was quickly reached to further cooperate on different tasks. In this cooperation, some of the future projects will be to create dynamic quality settings and to create specialized materials, like diamonds. An additional task will be to improve the material library that was implemented during this project, and make the materials editable by the user in real time.

## References


[1] ShapeDiver GmbH. www.shapediver.com. Accessed: 26-09-2017.

[2] Khronos Group. Webgl. www.khronos.org/webgl. Accessed: 26-09-2017.

[3] Javascript 3D library. three.js. threejs.org. Accessed: 26-09-2017.

[4] Lance Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12(3):270–274, August 1978.

[5] Randima Fernando. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[6] Morgan McGuire, Michael Mara, and David Luebke. Scalable ambient obscurance. In *Proceedings of the Fourth ACM SIGGRAPH*, EGGH-HPG'12, pages 97–103, Aire-la-Ville, Switzerland, 2012. Eurographics Association.

[7] Louis Bavoil and Miguel Sainz. Screen space ambient occlusion. *NVIDIA developer information*, 6, 2008.

[8] Brent Burley. Physically based shading at disney. *Walt Disney Animation Studios*, 2014.

[9] Brian Karis and Epic Games. Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice*, 2013.

[10] Ronen Basri and David W Jacobs. Lambertian reflectance and linear subspaces. *IEEE transactions on pattern analysis and machine intelligence*, 25(2):218–233, 2003.

[11] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 195–206. Eurographics Association, 2007.

[12] TS Trowbridge and KP Reitz. Average irregularity representation of a rough surface for ray reflection. *JOSA*, 65(5):531–536, 1975.

[13] Christophe Schlick. An inexpensive brdf model for physically-based rendering. In *Computer graphics forum*, volume 13, pages 233–246. Wiley Online Library, 1994.

[14] B Smith. Geometrical shadowing of a random rough surface. *IEEE transactions on antennas and propagation*, 15(5):668–671, 1967.