# An Evaluation of Deferred Shading Under Changing Conditions

Bart Postma and Tobias Isenberg

**Abstract**—Deferred shading has gained popularity in recent years, but some aspects about its performance are still unclear. In this work we make an initial evaluation of deferred shading's performance and evaluate it with respect to the number of light sources, the type of light sources, the illumination model, the complexity of the scene, and the influence range of the light sources. We compare measured performance values with each other and also compare them with the performance of traditional shading. The results show that in particular the number of light sources, the type of light sources, and the scene complexity are important factors that influence the performance of deferred shading.

**Index Terms**—Deferred shading, changing conditions, performance, real-time rendering.

## 1 Introduction

Deferred shading was first introduced in 1988 by Deering et al. [3]. Its basic principle lies in postponing the shading to a final 2D post-process for performance reasons. Deferred shading has gained in popularity since it became feasible on consumer graphics cards, but a rigorous evaluation about its performance is still missing. The choice between deferred shading and traditional shading, however, can have a fundamental influence on the performance, either positive or negative. In this work we analyze the performance of deferred shading and how it behaves under changing conditions and we also compare the results with the performance of traditional shading.

Traditional shading renders the scene and shades it directly. With deferred shading, a first rendering pass renders the geometry attributes of the scene to a buffer without performing shading. Specifically, geometry attributes such as position, color, normal, material properties, etc. are rendered to a G-buffer [12]. When this is finished, the G-buffer contains a projection of the scene's geometry attributes observed from the current view point. In a second pass, the shading is performed as a 2D post-process by using the light contributions on the scene's projection and the geometry attributes stored in the G-buffer from the first pass. One of the advantages of this is that shading computations are only performed on those parts of the scene that are visible in the final image. Deferred shading, therefore, shades the absolute minimum of the scene, whereas traditional shading may shade parts of the scene that are not visible in the final image. The performance penalty of deferred shading comes from the requirement of first rendering the scene to a G-buffer and later retrieving geometry attributes from this buffer. This requires much bandwidth between the GPU and video memory. The shading quality of the two techniques is identical as long as sufficient precision is used for the G-buffer elements. In recent years, other techniques related to deferred shading have been developed, such as light indexed deferred lighting [14], light pre-pass rendering [4], and inferred lighting [8], each with its own advantages and disadvantages.

Hargreaves and Harris [7] promoted deferred shading by illustrating how to apply it on consumer graphics cards. Their work, however, remains rather general with respect to performance, only mentioning a few situations where it can lead to a performance increase. Other works [9, 13] discuss the role of deferred shading in two recent computer games. Computer games, however, have many application-specific optimizations that are not suitable for all applications. Their works are valuable though for possible optimizations and pitfalls of deferred shading. Some initial performance measurements are presented in [6], but this work is not focused on deferred shading's performance and the presented measurements are thus kept basic. Moreover, knowledge regarding the test scene and the light sources is limited.

We extend the work of the above mentioned sources by conducting a high-level evaluation of the performance of deferred shading. To be able to generalize the results we do not perform application-specific optimizations, but keep the results applicable to a broad range of applications. The reason behind this is that each individual application has many variables of influence, e. g., scene types, lighting situations, underlying hierarchies or tiling of geometry for quick culling, supported GPU types, etc., which makes a full evaluation extremely difficult. Therefore we restrict ourselves to a number of conditions that are present in almost all graphics applications and that have a direct influence on the performance. These conditions concern the number of light sources, the type of light source, the illumination model, the complexity of the scene, and the influence range of light sources.

## 2 Implementation

We developed an OpenGL implementation for both traditional and deferred shading. The scene and the light sources are stored in Vertex Buffer Objects (VBOs).

With traditional shading, the first rendering pass transforms the light positions to eye space with a vertex program because shading is performed in eye space (Fig. 1). The eye-space positions are stored in a Texture Buffer Object (TBO). The second pass processes the scene and shades it by using the light positions stored in the TBO. In the case of spot light sources a second TBO is added, containing the spot directions.

In many graphics applications the bottleneck is the processing of the fragments, therefore traditional shading is often extended with the $z$ pre-pass technique [10]. With $z$ pre-pass we

*Bart Postma and Tobias Isenberg are with the University of Groningen, E-mail: b.j.j.postma@rug.nl and isenberg@cs.rug.nl , resp.*

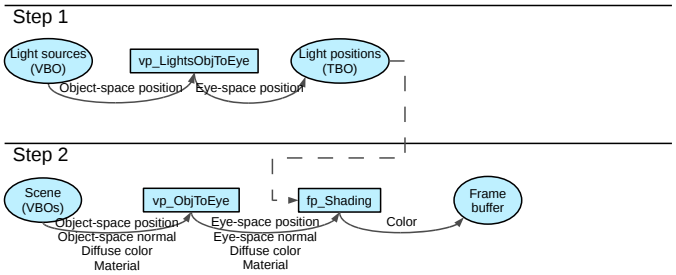


Fig. 1: Implementation of traditional shading.

render the scene in a first pass as fast as possible to fill the $z$-buffer. A second rendering pass renders the scene as usual, but the information in the $z$-buffer is used to reject fragments before processing them with a fragment program. Since $z$ pre-pass is a widely used technique and it has some resemblance to deferred shading by saving computations on occluded objects, we also take it into account in our evaluation.

With deferred shading, the first pass renders the scene to the textures of a Frame Buffer Object (FBO) which functions as the G-buffer (Fig. 2). The second pass performs ambient shading. The final pass processes the light contributions to the scene and computes the corresponding shading. By enabling additive blending in the final pass, the shading is added to the ambiently shaded scene and overlapping shading contributions are added to each other. Light contribution is determined with a geometry program. Depending on the type of the light source and its parameters, this program constructs the corresponding light volume. The light volume is projected onto the screen and rasterized into fragments, which are used by a fragment program to retrieve the corresponding geometry attributes from the G-buffer and perform the shading.

Constructing relatively complex light volumes in a geometry program comes at a cost, therefore it is worthwhile to optimize this step. For a spot light source, with a cone representing its influence range, we construct a four-sided pyramid that bounds the cone. This requires the construction of only four triangles, the base of the pyramid does not need to be constructed since it is only the pyramid's projection we are interested in. For a point light source with local spherical influence, a billboard (two triangles) can bound the sphere's projection from all view points. For a light source with global influence, a full-screen

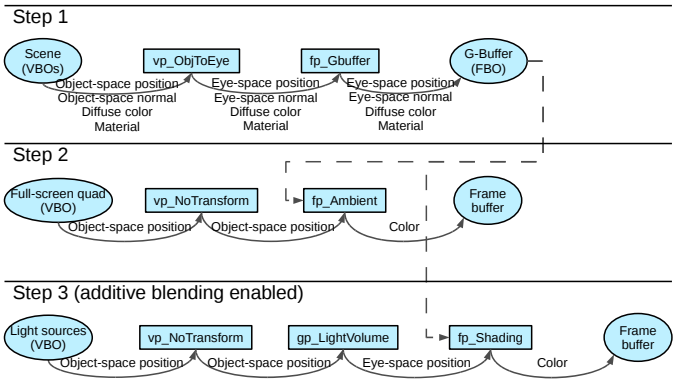


Fig. 2: Implementation of deferred shading.

| Component | Texture 1 | Texture 2 | Texture 3 |
|---|---|---|---|
| Red | Diffuse color Red | Position X | Normal X |
| Green | Diffuse color Green | Position Y | Normal Y |
| Blue | Diffuse color Blue | Position Z | Normal Z |
| Alpha | Ambient factor | Specular factor | Shininess |

Table 1: Organization of the G-buffer.

quad is constructed, denoting its influence on the entire scene.

A second optimization is to minimize the number of G-buffer textures. Instead of having a separate texture for each geometry attribute we organize the components of the G-buffer textures more efficiently. Shishkovtsov [13] presents a number of organizations for this purpose, together with their advantages and disadvantages. We use an organization optimized for speed and not memory by storing all geometry attributes in three textures (Table 1). The components are stored as 16 bit floating-point numbers, which is found in the majority of applications that use deferred shading.

## 3 Evaluation Approach

All the tests of our evaluation measure the framerate and vary one condition, while keeping the other conditions unchanged. For a fair comparison between traditional and deferred shading we have to avoid the CPU being the bottleneck. Therefore, both shading techniques run solely on the GPU. The CPU is only used for setting up the rendering and for handling user interaction. In addition, we animate the light positions and the view point to mitigate caching optimizations.

Two well-known illumination models are used, i. e., Phong [11] and Cook-Torrance [2], where the first is probably the most widely used model and the latter is a relatively expensive model. Initially, we also included the Blinn-Phong [1] and the Gooch [5] illumination models in our evaluation, but since they did not result in any noticeably differences compared to the results with Phong illumination we are not including them in our discussion.

The most common types of light sources are spot, point, and directional light sources, where the first two have a local influence range and the latter has a global influence range. We use the former two, but since directional light sources do not have a position to be animated they are replaced by point light sources with a global influence range. From a performance perspective this change is negligible, the only difference is that distance attenuation is performed and the light direction is computed instead of being given.

The number of light sources is varied from no light sources to 500 light sources. This number proves to be high enough to see how traditional and deferred shading scale with the number of light sources.

The scene complexity is expressed in number of generated fragments. Early testing of our application revealed that the performance is bound by the number of fragments being processed. Therefore, we choose to express the scene complexity in number of fragments instead of number of vertices. The test starts with one object and additional objects are added until the scene contains all objects.

The impact of the influence range of light sources is mea-

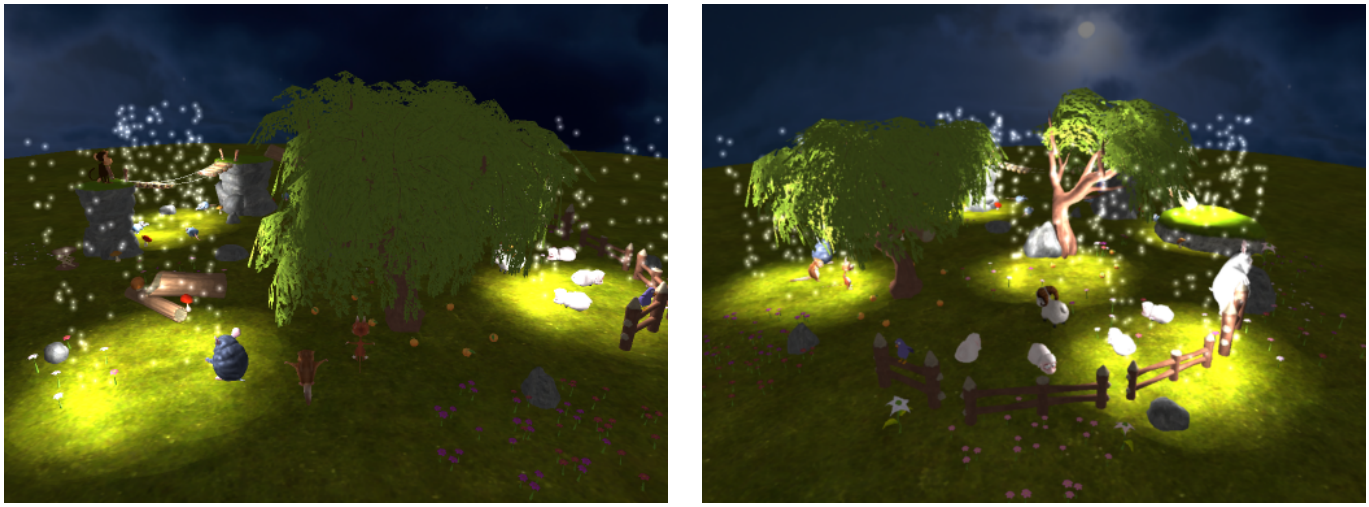

Fig. 3: Test scene observed from two view points. The left image contains 300 light sources and the right image 500 light sources.

sured by gradually increasing their influence range, from a relatively small influence range to an influence range that affects almost all objects in the scene.

For a proper evaluation we are required to reproduce conditions that can reasonably be expected in today's graphics applications. Consequently, we have to make a number of assumptions regarding the test scene and the light sources. The test scene has to meet a sufficiently high complexity level for the results to be relevant. Our test scene consists of $1.2 \cdot 10^5$ vertices and uses a total of 23 textures, ten of them are of size $512 \times 512$, another ten of size $1024 \times 1024$, and the remaining three have sizes of $360 \times 360$, $1024 \times 256$, and $2018 \times 640$ (Fig. 3). We make three assumptions regarding the light sources. Firstly, the impact of scene complexity is evaluated separately with 15 spot light sources, 15 point light sources with local influence, and 3 point light sources with global influence. Secondly, the impact of the influence range of light sources is evaluated with 5 point light sources. Lastly, a light source influences several objects most of the time. For this last assumption we use the fact that objects in the test scene have an $x,z$-position within a radius of 100 from the scene's center (in object space) and the tallest object does not exceed a height of $y = 50$. Hence we limit $x,z$-positions of the light sources to a radius of 100 from the scene's center and distribute their height uniformly between a relatively low height of $y = 3$ and $y = 50$. Additionally, point light sources with local influence receive an influence radius of size 20 and spot light sources are directed downward and receive a cut-off angle of 45°.

The evaluation is performed at a resolution of $1280 \times 1024$ and is conducted under Linux on a machine with a dual-processor AMD Opteron 280 2.4 GHz dual-core, 8 GB RAM, and an Nvidia Geforce GTX 285 graphics card. We also tested on several machines with older hardware, but the results show fairly identical behavior across all machines.

## 4 Results

We present the results based on the assumptions identified in Section 3. In the first test, the *number of light sources* is varied from no light sources to 500 light sources. The most remarkable observation of this test is the relative insensitivity of deferred shading to the number of light sources compared to traditional shading (Fig. 4). This is particularly true for light sources with local influence, i. e., spot light sources and point light sources with local influence. Adding more light sources with traditional shading means that every fragment is processed by an increased number of light sources, whereas deferred shading only processes the fragments and light contributions that are visible in the final image. Secondly, more light sources also means a a higher probability of spending shading computations on occluded objects with traditional shading. This is mitigated with $z$ pre-pass, but the results do not change significantly. The cost of constructing the G-buffer makes deferred shading start with a lower framerate than traditional shading, and it requires approximately five light sources before the cost outweighs the benefits, depending on the light source type and illumination model. With light sources that have a global influence there is no noticeable advantage of deferred shading over traditional shading. Global influence requires frequent access of the G-buffer, thereby utilizing much bandwidth, limiting deferred shading's performance significantly.

The results presenting the influence of *scene complexity* have two things in common: they all show a linear performance decrease with respect to the complexity and they all have several rather abrupt decreases in framerate (Fig. 5). The linear performance decrease is a consequence of the application being bound by the number of fragments, adding more fragments to the scene lowers the performance proportionally. The abrupt decreases can be explained by looking more closely to when and how many objects are drawn in the test. It seems that the abrupt decreases occur when objects are rendered consisting of relatively many vertices. Although the application is bound by the number of fragments, it seems that the number of vertices is not entirely negligible. Note that this test is not suitable for comparing the absolute framerates of the two shading techniques, but for comparing how the two shading techniques scale. The reason for this is that the number of light sources is fixed and the

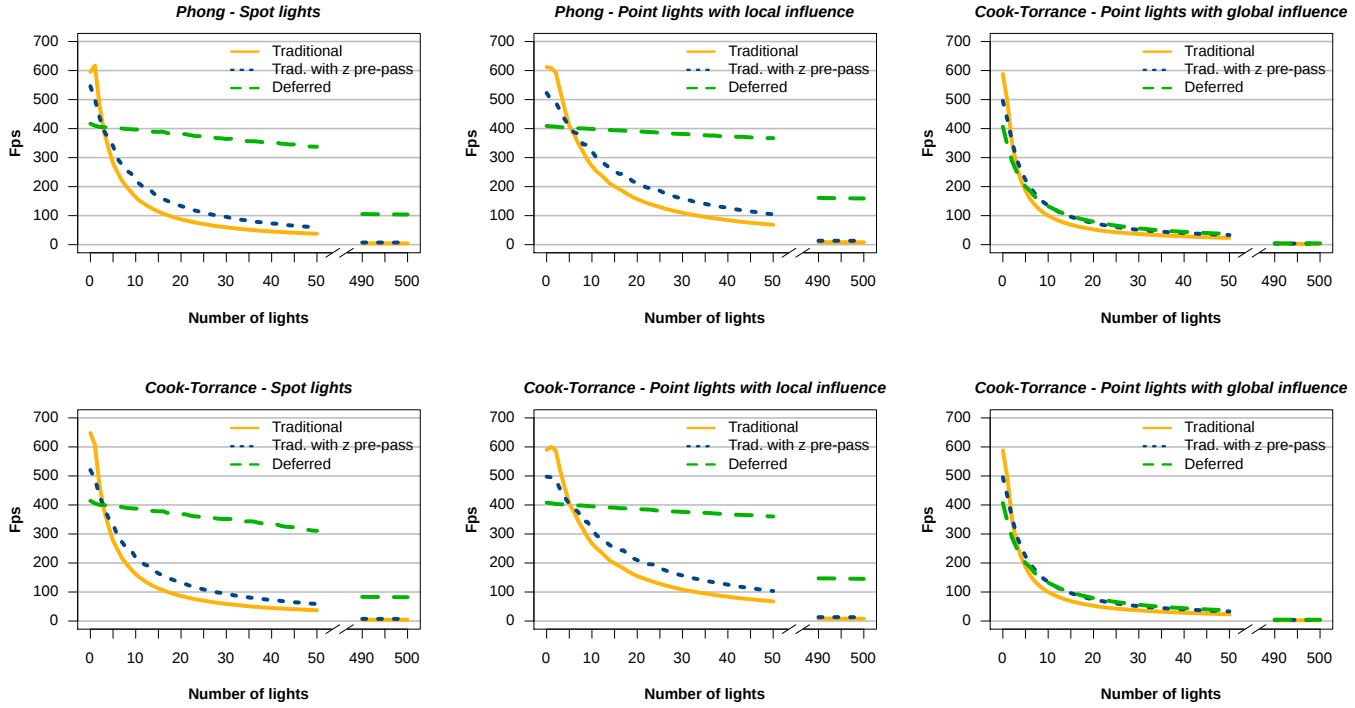


Fig. 4: Frame rate versus the number of light sources.

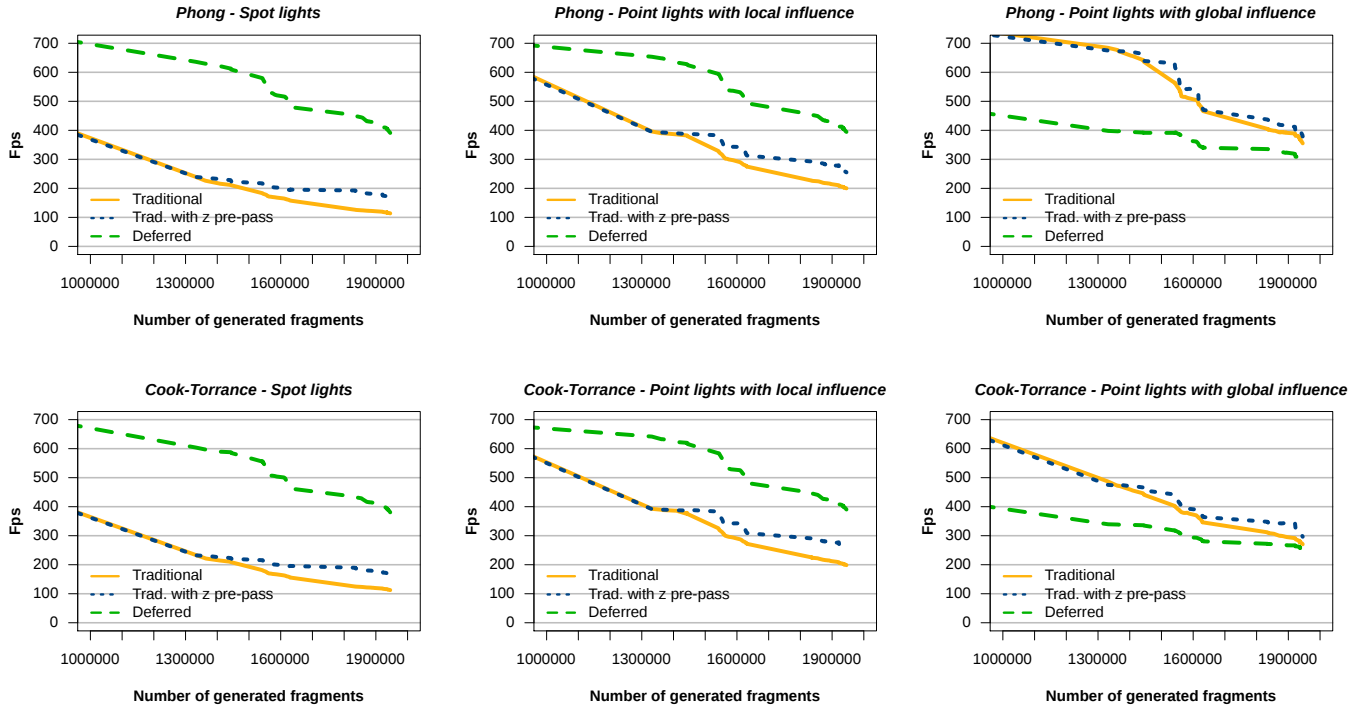


Fig. 5: Frame rate versus the number of generated fragments.

first test already revealed that one shading technique performs better than the other with a certain number of light sources. The same also holds for the next test.

The test relating framerate and light source's *influence range* is performed with five point light sources, starting with a relatively small influence radius of size 10 and ending with a relatively large influence radius of size 120. Most objects in the scene reside close to the ground, so increasing the influence radius increases the number of affected fragments more or less quadratically, therefore we might expect that the performance

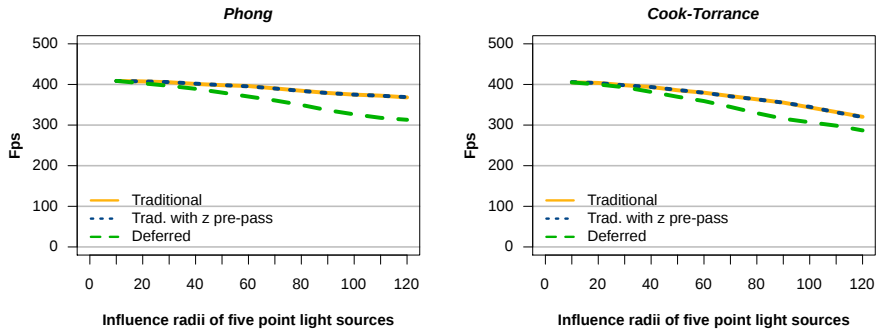


Fig. 6: Frame rate versus the influence radii of five point light sources.

decreases quadratically. There are two reason why the results show a decrease which is less than quadratic (Fig. 6). First, the shading is not a significant bottleneck with five light sources and relatively small influence radii. Second, not all fragments in the influence radius are shaded. Unnecessary shading is mitigated by only shading fragments that are facing towards a light source and by culling procedures of the graphics drivers. Traditional shading with and without $z$ pre-pass have nearly identical results in this test, which is due to the fact that both techniques have nearly identical results with five light sources (Fig. 4). On other tested machines $z$ pre-pass has a small advantage in this test.

Finally, we consider the influence of the *illumination model* (Figures 4–6). Here we can observe that deferred shading allows more expensive illumination models to be used, because its shading stage is cheaper compared to traditional shading. Even an illumination model such as Cook-Torrance may be used, which is usually considered to be too expensive.

## 5 Conclusions

The results show that both shading techniques have an ideal situation for their usage. For situations where there are not much more than approximately five light sources and the scene complexity is relatively low, traditional shading is well-suited. Whether or not an application should use $z$ pre-pass depends on its bottleneck. If it is the processing of the fragments, then $z$ pre-pass most likely gives performance benefits, otherwise it most likely does not. Deferred shading is a good choice for situations where there are many light sources, the light sources have a local influence, and the scene is relatively complex.

The difficulty in choosing arises when the situation is not known beforehand. The results show that deferred shading's performance is less sensitive to the tested conditions than traditional shading's performance, making deferred shading a safer choice in that case.

The presented results and conclusions are based on a high-level evaluation of both shading techniques. Application-specific optimizations, however, can sometimes give significant improvements. Also, both techniques require a different approach for handling effects such as anti-aliasing, transparency, shadows, etc., which can be a factor of influence in embracing or rejecting one of the two. Each application has its own properties and requirements and each application should be examined on its own terms. This work provides an initial step in choosing between the two shading techniques and gives insight in their performance at a high level.

## References

[1] J. F. Blinn. Models of Light Reflection for Computer Synthesized Pictures. *ACM SIGGRAPH Computer Graphics*, 11(2):192–198, Summer 1977. doi: 10.1145/965141.563893

[2] R. L. Cook and K. E. Torrance. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics*, 1(1):7–24, Jan. 1982. doi: 10.1145/357290.357293

[3] M. Deering, S. Winner, B. Schediwy, C. Duffy, and N. Hunt. The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics. In *Proc. SIGGRAPH*, pp. 21–30, New York, 1988. ACM. doi: 10.1145/54852.378468

[4] W. Engel. Designing a Renderer for Multiple Lights – The Light Pre-Pass Renderer. In *Shader X7*, chapter 8.5. Charles River Media, 2009.

[5] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A Non-Photorealistic Lighting Model For Automatic Technical Illustration. In *Proc. SIGGRAPH*, pp. 447–452, New York, 1998. ACM. doi: 10.1145/280814.280950

[6] L. Gruber. DeShade: A Deferred Shading Frame Work for Complex Lighting. Master's thesis, Graz University of Technology, 2008.

[7] S. Hargreaves and M. Harris. Deferred Shading. Technical report, Nvidia, 2004.

[8] S. Kircher and A. Lawrance. Inferred Lighting: Fast Dynamic Lighting and Shadows for Opaque and Translucent Objects. In *Sandbox: Proc. ACM SIGGRAPH Symp. on Video Games*, pp. 39–45, New York, 2009. ACM. doi: 10.1145/1581073.1581080

[9] R. Koonce. Deferred Shading in Tabula Rasa. In *GPU Gems 3*, pp. 429–457. Addison-Wesley Professional, 2007.

[10] E. Persson. Depth In-depth. Technical report, ATI, 2007.

[11] B. T. Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, June 1975. doi: 10.1145/360825.360839

[12] T. Saito and T. Takahashi. Comprehensible Rendering of 3-D Shapes. In *Proc. SIGGRAPH*, pp. 197–206, New York, 1990. ACM. doi: 10.1145/97880.97901

[13] O. Shishkovtsov. Deferred Shading in S.T.A.L.K.E.R. In *GPU Gems 2*, pp. 143–165. Addison-Wesley Professional, 2005.

[14] D. Trebilco. Light Indexed Deferred Lighting. Technical report, 2008.