Dreamspace

D4.1.1

Approaches for Real-Time Ray Tracing and Lighting Simulation

Deliverable due date: 30 June 2014 (M9)

Actual submission date: 30 June 2014

Re-submission date: 05 Jan 2015

Project no: FP7 - 61005

Project start date: 01.10.13

Lead contractor: The Foundry

Project ref. no.	FP7 - 61005
Project acronym	DREAMSPACE
Project full title	Dreamspace: A Platform and Tools for Collaborative Virtual Production
Security (distribution level)	CO
Contractual date of delivery	Month 9, 30.06.2014
Actual date of delivery	M16, 5th January 2015
Deliverable number	D4.1.1
Deliverable name	Approaches for Real-Time Ray Tracing and Lighting Simulation
Type	Report
Status & version	Final, 1.3
Number of pages	17
WP / Task responsible	Philipp Slusallek, DFKI
Other contributors	
Author(s)	Philipp Slusallek, DFKI
Internal Reviewer	Philippe Bekaert, iMinds Richard Membarth, DFKI
EC Project Officer	Alina Senn

DOCUMENT HISTORY

Version	Date	Reason of change
1.0	15-06-2014	Document created by Philipp Slusallek
1.3	08-12-2014	Update title; minor changes in response to Annual Review

Inhalt

1. INTRODUCTION 4

2. RENDERING AND LIGHTING SIMULATION 4

3. RAY TRACING 5

4. GLOBAL ILLUMINATION ALGORITHMS AND THEIR EFFICIENT IMPLEMENTATIONS . 6

5. SOFTWARE SYSTEMS AND ARCHITECTURES FOR REAL-TIME RAY TRACING 10

6. CONSEQUENCES FOR DREAMSPACE 11

7. CONCLUSIONS 11

8. LIST OF SELECTED PAPERS 12

9. REFERENCES 13

1. Introduction

This deliverable gives an overview of real-time ray tracing and lighting simulation algorithms. It consists of this document and ideally an appendix of the key publications that go into more details about the algorithms that are most relevant to the goals of the project.

However, due to copyright issues this appendix has been replaced by a list of links to online versions of these papers. So, we ask the reader of this document to please use these links to download versions of these papers.

2. Rendering and Lighting Simulation

A key aspect of computer graphics is the rendering of images from descriptions of 3D scenes. Given the geometry of objects in the scene, the optical properties of surfaces and volumes (i.e. transmission, reflection, and scattering of incoming light), the location and emission information for primary light sources, and finally the location and properties of a virtual camera, rendering computes the light field incident on the virtual film in the camera – an image.

Mathematically this corresponds to solving the so-called "Rendering Equation" [Kajiya 86], which computes the light (radiance) at each point visible from the camera as:

$$L(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega f_r(\omega_i, x, \omega_o) L_i(x, \omega_i) \cos(\theta_i)\, d\omega_i$$

This equation describes the light distributed from the surface point x in direction ω_o as the sum of the light L_e emitted from the surface itself (non-zero only for light sources) plus the scattered light. The scattered light is given by the integral over all incident light L_i from all directions in the sphere of direction Ω at x weighted by the cosine between the normal and the incident direction θ_i. For each incoming direction ω_i at point x we accumulate its contribution to the outgoing direction ω_o via the bidirectional reflectance distribution function (BRDF) f_r. The BRDF captures the optical material properties, such as diffuse and specular reflection and refraction.

The key challenge is that all incoming light L_i at some point x is actually the outgoing light from some other point y in the scene visible from x (possibly modulated by the intervening participating volume, ignored in the following). Thus the above equation couples the light field at every point in the scene to the light field of every other (visible) point. This is known as *global illumination computation*. Mathematically this corresponds to solving a Fredholm-Equation of the Second Kind, which is an infinite dimensional, recursive integral equation that for rendering often contains many discontinuities and singularities, making the solution for this equation a real challenge.

Historically, rendering started with simple *local illumination* methods that ignore all interactions of light with other parts of the scene and thus does not simulate shadows or any global lighting effects, such as indirect illumination, color bleeding, and smooth shadows. This is the simplest approach and is the basis for all hardware implementations in today's GPUs. More advanced methods perform *direct illumination* by additionally taking the interaction of light only on its direct way from a light source to the illuminated point into account (shadows), but ignore all other global effects. GPUs already require advanced techniques to even approximate direct lighting effects (shadow maps, shadow volumes, and related methods) but cannot accurately compute it.

Accurate and full global lighting simulations today are almost exclusively computed through Monte-Carlo algorithms that use a randomized sampling approach to solve the above integral equation. These methods require the tracing of many random ray paths through the scene in order to determine the mutual visibility between subsequent sampling points. The most practical solution techniques today are *path tracing* and *bidirectional path tracing* (see below).

After many years in which simple direct lighting techniques have been used almost exclusively in film and movie industry (e.g. Pixar's Renderman), the industry has now switched to almost exclusively using physically-based Monte-Carlo approaches. Because of their algorithmic complexity these methods require significant more computation times, often reaching hours for even a single image. This makes real-time realistic rendering with these lighting simulation methods a significant challenge – which we are addressing in Dreamspace through a clever combination of algorithmic improvements, new compiler technology for easily writing programs that exploit the hardware capabilities of today's processors, as well as scalable systems that can harness the combined computational resources of many compute nodes in a fast network through suitable high-performance middleware. The algorithmic basis for this work is discussed in the following.

3. Ray Tracing

Ray tracing is famous for its ability to generate high-quality images but is also well-known for long rendering times due to its high computational cost. This cost is due to the need to traverse a scene with many rays, intersecting each ray with geometric objects along the way, shading the visible surface samples, and finally sending the resulting pixels to the screen. Due to the cost associated with ray tracing the technique has in the past been viewed exclusively as an off-line technique for cases where image quality matters more than rendering speed. Despite considerable advances in the last 15 years real-time ray tracing (see below) is still highly challenging.

However, ray tracing offers a considerable number of advantages over other rendering techniques. Basic ray casting, i.e. sampling the scene with individual rays, is a fundamental task that is the core of a large number of algorithms in computer graphics, including visibility computations, lighting simulation, line-of-sight algorithms, collision detection, and many more. It is also used in many other disciplines for example to simulate propagation of radio waves, neutron transport, audio simulations, and diffusion.

Ray tracing goes back to very basic ray casting techniques by Appel in 1968 [Appel 68] but was first fully described by Whitted in 1980 [Whitted 80], which captures the full backward tracing of rays from the camera into the scene to find the visible points. From those "shadow rays" are cast to light sources thus accurately calculating the direct illumination of these points. Global illumination is not fully addressed except for light through mirror reflection of refraction in transparent surfaces. These are computed by recursively tracing rays in the respective directions. This basic ray tracing model is still in wide use today for rendering.

Many early attempts had been made at accelerating ray tracing to real-time speeds but most have failed, such that in the 1990ies the research community had mostly given up on the attempt. A first glimpse of what might be possible was given by Parker et al. [Parker 98], showing that real-time speeds can be achieved on large supercomputer of the time.

A breakthrough happened in 2001 when Wald et al. from Saarland University [Wald 01a, now over 500 citations] showed that huge speedups (eventually by a factor of 30x and more) could be obtained even on consumer CPUs at the time. The breakthrough consisted of three components: A novel packet-based approach to ray tracing and suitable parallel handling of acceleration structures (specifically kd-trees, later also Bounding Volume Hierarchies and others), use of SIMD instruction (e.g. Intel's SSE) that could take advantage of working on multiple rays from the packets simultaneously, and finally clever implementation techniques for optimizing the computation in general.

In the following years an increasing number of papers were published that improved on the basic technology by Wald and applied it to various domains. From 2006 on a separate conference was formed ("IEEE Symposium on Interactive Ray Tracing"). Today this line of research is best represented in the "Eurographics/Siggraph High-Performance Graphics" conference that is a direct successor of that conference. The research community in this area is still highly active today.

In the following we give an overview of the research in real-time ray tracing (RTRT) mainly based on work by Dreamspace partners but also list some important related work.

Even in its first year RTRT was applied to large models as ray tracing scales particularly well (logarithmically) with scene size [Wald 01b]. This was later scaled and improved to out-of-core methods [Wald 04a], large CAD models [Dietrich 07a, Dietrich 07b], and to extremely large outdoor scenes in [Dietrich 05, Dietrich 06a, Dietrich 07c].

A first API for RTRT (OpenRT) was proposed by Dietrich et al. in 2003 [Dietrich 03], who also addressed the integration of RTRT into common scene graphs for easier use by applications [Dietrich 04, Rubinstein 09]. Another key improvement was the handling of dynamic scenes [Wald 03c], later improved by Günther [Günther 06a, Günther 06b] and others.

Other key work addressed the efficient building of acceleration structures that minimize the amount of work RTRT has to spend on finding intersections with the scene. This includes the key techniques of streaming construction [Popov 06], the use of spatial splits for improving the spatial index structures [Stich 09, Popov 09], and the use of (multi-level) grids specifically on GPUs [Kalojanov 09, Kalojanov 11].

RTRT was extended to volumetric objects by Marmitt [Marmitt 04, Wald 05a], to free-form surfaces [Benthin 04, Benthin 06a], and to point clouds [Wald 05b]. RTRT has been applied in many application areas, with especially significant success for the aerospace industry [Dietrich 06b, Dietrich 07a, Löffler 11] but also for the visualization of large-scale scenes [Marsalek 10]. One application that is specifically interesting in the Dreamspace context is the use of RTRT as a server-based technique [Repplinger 09].

A key development at the time was the design of hardware architectures for RTRT. Initial work was done by Schmittler et al. [Schmittler 02, Schmittler 03, Schmittler 04]. A key breakthrough on hardware support for RTRT was the Siggraph publication by Woop et al. [Woop 05] with its follow-up work [Woop 06a, Woop 06b]. RTRT was implemented and optimized for different processors (variants of x86, Itanium, etc.) and hardware architectures, like the Cell processor [Benthin 06b, Davidovic 11] and in particular GPUs [Popov 07a, Günther 07]. Specifically for GPU implementations there is significant related work by others.

4. Global Illumination Algorithms and Their Efficient Implementations

All of the above techniques are limited to direct illumination computations (only effects in the direct light path between light source and surface) plus perfect reflections and refractions. All other effects such as smooth shadow boundaries due to extended light sources, indirect illumination via other surfaces in the environment, and caustics due to light being focused via specularly reflecting or refracting surfaces is not taken into account. Traditionally, this has first been addressed by Distribution Tracing [Cook 84], which also enabled the simulation of depth-of-field, motion blur, anti-aliasing, and other effects by randomly sampling the respective dimensions with a number of rays.

However, Distribution Tracing suffers from the effect that exponentially more ray need to be recursively traced for an exponentially diminishing effect on the final image. This was solved by Kajiya [Kajiya 86] with what we now know as Monte-Carlo (MC) integration techniques. Instead of sending many rays randomly sampling each of the many dimensions (pixel area, lens area, shutter time interval, light source area, etc.), it selects one of the dimensions only, generating a single ray path and then averages over many such paths. These ray paths constitute the secondary estimators in Monte-Carlo integration techniques.

Since Monte-Carlo techniques rely on random samples they suffer from noise and relatively slow convergence properties ($O(\frac{1}{\sqrt{N}})$). For rendering, the first issue is mainly addressed by importance sampling techniques (see specific techniques below) while the latter is mainly addressed through (jittered) stratification as well as Quasi-Monte-Carlo (QMC) that both try to distribute samples evenly across the different integration domains. Specifically, QMC can achieve an asymptotically better convergences rate but mainly in areas without discontinuities and singularities, which – unfortunately – appear often in rendering problems.

Monte-Carlo techniques come in four flavors, (i) path tracing, (ii) photon tracing/mapping, (iii) bidirectional methods, and (iv) Metropolis sampling. *Path tracing* is the basic method introduced already by Kajiya [Kajiya

86]. It traces a path of rays from the camera recursively into the scene trying to connect it directly to a randomly selected point on a light source by sending "shadow rays" from each sample point in the scene. This approach can fail or converge only slowly in cases where connections to the light source are difficult to find, such as by indirect illumination through small opening (e.g. a small window) or via specular objects (e.g. a mirror) where it is essentially impossible to find where to send a ray in order for it to hit the light source.

Photon Mapping [Jensen 96, Jensen 01] avoids this issue by randomly tracing photon using the same MC techniques from all light sources into the scene and depositing them of surfaces hit along the path. In a second, during normal (path) tracing, *density estimation* is used at each surface sample to determine the light energy deposited in the vicinity. This approach is particularly well suited for caustic effects that are hard to render with MC techniques alone. It is thus a complementary technique. The fact that density estimation is a technique that has a completely different mathematical basis and could not be integrated well with MC rendering has been causing significant inefficiencies from the beginning that were only recently being addressed (see below, [Georgiev 12b]).

Bidirectional methods, like bidirectional MC path tracing, address most of the above concerns, except some caustic and glossy effects. They start by randomly sampling paths into the scene from both the camera and the light sources, trying to connect the paths between all the generated samples on surfaces. This greatly increased the probability of finding good connected paths between the camera and light sources and consequently reduces noise in the image. A key component of these bidirectional methods (but not limited to them) is the use of Multiple Importance Sampling (MIS) for efficiently combining the different estimators that can be created by combining sub-paths started from both the camera and lights. In the past, such bidirectional paths where not able to sample Photon Mapping effects well, but again this has been addressed by the Vertex Connection and Merging (VCM) approach [Georgiev 12b].

Finally, *Metropolis Light Transport* [Veach 97] uses a special approach to deal with notoriously difficult cases of light transport. The approach starts with randomly selected paths. It then generates new paths using random mutations of an existing path and randomly accepts them based on the relative light contribution and the probability of the new path versus the previous one. This approach can deal with difficult light transport cases by exploring a local neighborhood of a good light transport path. However, it has turned out to be difficult to adjust the mutation strategies in suitable ways. As a result, the metropolis light transport approach is not commonly used much today.

Finally, it is important to note that there are two other and very different approaches to computing global illuminations: *Finite element methods*, like Radiosity, have been highly popular in the beginning of physically based lighting simulation research. However, since they are based on basis functions they are often sources of significant bias (shadow/light leaks) and cannot deal well with discontinuities/singularities nor with higher-dimensional effects. Secondly, there is a whole class of *rasterization based methods* that are optimized for execution on GPUs. Because of the design of GPUs, these methods have to make (sometimes strong) approximations and assumptions in order to be implementable on GPUs. In cases where these assumptions hold, these methods easily reach performance levels of dozens if not hundreds of frames per second. However, they easily break down and show artifacts in cases where this is not the case. These methods are well accepted for games, but not as general lighting simulation methods. They are normally not used in the film and movie industry where high realism, reliability, and artifact-free computations are of high importance.

In the following we review some of the key publications in the area, again focusing on publications from the Dreamspace consortium with selected additions of key other papers.

Early real-time lighting simulation: Quite early, RTRT was already being used for also computing global illumination effects. Initial publications started already in 2002 with Wald et al. [Wald 02]. Other early publications have been [Benthin 03, Wald 03a]. Also caustic effects were addressed early on by a Photon Mapping based solution in [Günther 04, Wald 04b] that also addressed the scalability issues inherent in that solution via interleaved sampling.

Glossy Illumination: A key problem in lighting simulation still is the efficient handling of glossy effects that appear in many situations. They suffer from the fact that they fall in between the diffuse and purely specular reflection and are hard to sample efficiently. Davidovic et al. [Davidovic 10] devised a new algorithmic approach to improve the situation using a combination of local and global lights. The paper was published at the prestigious Siggraph conference in 2010.

Adaptive Importance Sampling: In parallel Georgiev developed a highly efficient approach to adaptively importance sample the large numbers of light sources generated by Many-Light Techniques [Georgiev 10]. It presents a simple and practical algorithm for importance sampling virtual point lights (VPLs) and is suitable for multi-pass rendering. During VPL distribution, a Russian roulette decision accepts VPLs proportionally to their estimated contribution to the final image. As a result, more VPLs are concentrated in areas that illuminate the visible parts of the scene, at the cost of a negligible performance overhead in the preprocessing phase. As VPLs are sampled independently and proportionally to their camera importance, the algorithm is trivial to parallelize and remains efficient for low sampling rates. The paper showed that this sampling scheme is well suited to both well illuminated scenes as well as for difficult visibility conditions. Moreover, in contrast to bidirectional and Metropolis VPL sampling techniques, the algorithm is fast and very simple to implement, and uses a single Monte Carlo sampler, making it easier to maintain good stratification.

Importance Caching: This work was extended in [Georgiev 12a] published at Eurographics. It identifies the key to good performance in carefully selecting the costly integration samples, which is usually achieved via importance sampling. Unfortunately, visibility is difficult to factor into this importance distribution, which can greatly increase variance in highly occluded scenes with complex illumination. The paper therefore introduces the important concept of *Importance Caching* – a novel approach that selects those samples with a distribution that includes visibility, while maintaining efficiency by exploiting illumination smoothness. At a sparse set of locations in the scene, it constructs and caches several types of probability distributions with respect to a set of virtual point lights (VPLs), which notably include visibility. Each distribution type is optimized for specific lighting conditions. For every shading point, the approach then borrows the distributions from nearby cached locations and uses them for improved VPL sampling, avoiding additional bias. A *novel multiple importance sampling framework* finally combines the many estimators. In highly occluded scenes, where visibility is a major source of variance in the incident radiance, this approach can reduce variance by more than an order of magnitude. Even in such complex scenes this allows to obtain accurate and low noise previews with full global illumination in a couple of seconds on a single mid-range CPU. This approach is well suited for the goal we have in Dreamspace.

Vertex Connection and Merging (VCM): A huge step forward also has been the subsequent work by Georgiev later in the same year [Georgiev 12b]. He addressed the problem that developing robust light transport simulation algorithms that are capable of dealing with arbitrary input scenes remains highly difficult. Although efficient global illumination algorithms exist, an acceptable approximation error in a reasonable amount of time is usually only achieved for specific types of input scenes. Specifically, the use of Photon Mapping for caustic-like effects and the predominant Monte-Carlo integration had remained separate techniques that we hard to integrate – both mathematically and from the point of view of efficient implementations.

To address this problem, Georgiev presents a reformulation of the photon mapping as a bidirectional path sampling technique for Monte Carlo light transport simulation. The benefit of this new formulation is twofold. First, it makes it possible, for the first time, to explain in a formal manner the relative efficiency of photon mapping and bidirectional path tracing, which have so far been considered conceptually incompatible solutions to the light transport problem. Second, it allows for a seamless integration of the two methods into a more robust combined rendering algorithm via multiple importance sampling. A progressive version of this algorithm is consistent and efficiently handles a wide variety of lighting conditions, ranging from direct illumination, diffuse and glossy inter-reflections, to specular-diffuse-specular light transport. Our analysis shows that this algorithm inherits the high asymptotic performance from bidirectional path tracing for most light path types, while benefiting from the efficiency of photon mapping for specular-diffuse-specular lighting effects.

It is important to note that since the publication of this paper, that the VCM technique has been widely adopted in the industry. Besides products such as V-Ray, Corona, and LuxRender, Pixar has publically announced their use of VCM in their completely redesigned Renderman V19 product that is the main work horse of many studios in the industry. VCM is prominently mentioned in both the announcement as well as the FAQ (available online at http://renderman.pixar.com/view/DP25846 and http://renderman.pixar.com/-view/25919).

This paper does not yet achieve real-time performance (but see [Davidovic 14] below) – it rather focuses on the algorithmic improvements and on integrating all major effects into a single and rather simple algorithmic formulation.

Joint Path Sampling: In follow-up work, the previous method was extended to address also the much more complex issues of computing lighting simulation in participating media, such as fog. As stated above, it is central to all Monte Carlo-based rendering algorithms to construct light transport paths from the light sources to the eye. Existing rendering approaches sample and create new path vertices incrementally along a path when constructing them. The resulting probability density is thus a product of the conditional densities of each local sampling step, constructed without explicit control over the form of the final joint distribution of the complete path. This paper shows why current incremental construction schemes often lead to high variance in the presence of participating media, and reveal that such approaches are an unnecessary legacy inherited from traditional surface-based rendering algorithms.

The paper then devises joint importance sampling of path vertices in participating media to construct paths that explicitly account for the *product of all scattering and geometry terms along a sequence of vertices* instead of just locally at a single vertex. This leads to a number of practical importance sampling routines to explicitly construct single- and double-scattering sub-paths in anisotropically scattering media. It demonstrates the benefit of the new sampling techniques, integrating them into several path-based rendering algorithms such as path tracing, bidirectional path tracing, and many-light methods. It also uses the sampling routines to generalize deterministic shadow connections to connection sub-paths consisting of two or three random decisions, for efficiently simulating higher-order multiple scattering. The resulting algorithms significantly reduce noise and increase performance in renderings with both isotropic and highly anisotropic, low-order scattering. However, these methods are not yet suitable for real-time applications.

Progressive Light Transport Simulation on the GPU: Finally, in a paper to be published at Siggraph this year, Davidovic integrated many of the above algorithms into a coherent and consistent architecture that runs on highly parallel, many-core hardware architectures (GPUs). GPUs recently became general enough to enable implementation of a variety of physically-based light transport algorithms. However, the efficiency of these GPU implementations has received relatively little attention in the research literature and no systematic study on the topic existed previously. In particular non that compared the different implementations in a truly comparable way.

The main contribution of that paper is to fill that gap and give a comprehensive and in-depth investigation of the efficiency of the GPU implementation of a number of the above light transport simulation algorithms. In addition, the paper develops several improvements over the state-of-the-art. In particular, the Light Vertex Cache, a new approach to mapping connections of sub-path vertices in Bidirectional Path Tracing on the GPU, outperforms existing implementations by 30-60%. The paper also contains a first GPU implementation of the above Vertex Connection and Merging algorithm [Georgiev 12b], showing that even relatively complex light transport algorithms can be efficiently mapped onto the GPU. With the implementation of many of the state-of-the-art algorithms within a single system available, the paper presents a unique direct comparison and analysis of their relative performance benefits.

While the results of the paper are limited to GPU implementations, many of the insights can be used in other context as well. As a result, we plan to use the implementations in this paper as a basis for the work in Dreamspace. The conceptual abstractions developed in Dreamspace using AnyDSL needs to be compatible with requirements formulated here and the algorithms need to be adjusted to these abstractions – such that they do not lose their efficiency when mapped to GPUs.

5. Software Systems and Architectures for Real-Time Ray Tracing

We currently have two major implementations for RTRT that are being used on a larger scale: The Optix System developed by Nvidia [Parker 10] is an API and a library of ray tracing algorithms targeting mainly Nvidia GPUs. The system uses a special compiler that combines snippets of code into a high-performance RTRT code. Optix is a proprietary system by Nvidia.

The Embree system [Woop14] is mainly a collection of highly optimized kernels for x86 CPUs and the Intel MIC architecture developed by Intel and made available as Open Source. The system has been adopted by a number of commercial companies, also including the movie industry. The Embree software distribution contains a simple path-tracer used to demonstrate its rendering performance but not meant to be a full rendering system. We are using Embree within the Dreamspace project as a remote renderer streaming its result to Web-based clients.

Compiler technology has been used also to adapt sequential shader code that is easier to write for non-expert developers (content creators) to be compatible with the highly optimized and often parallel kernels used in RTRT. Karrenberg developed the AnySL system [Karrenberg 10] that uses a novel Whole-Function Vectorization approach to generate shading code that fits nicely into existing renderers. Its flexible use of different interface libraries with functions that eventually are all inlined into a single highly optimized piece of code has been instrumental for this purpose. Its vectorization approach has been copied by many others but until today it still outperforms several of the commercial systems.

A key paper on software design for RTRT has been the RTFact architecture by Georgiev et al. [Georgiev 08]. For the first time this paper showed that high-level programming approaches can be used to allow developers to easily formulate the algorithms for RTRT (as well as other fields) while still being able to automatically map these algorithms to highly efficient code to be executed on different hardware platforms, achieving performance results within 10% of the expensive, hand-coded optimizations.

RTFact, however, used C++ Template Metaprogramming for its implementation, which is effectively a functional programming language layered on top of C++. It has a particularly weird syntax that is difficult to write end even more difficult to read, is not type-safe, and is hard to debug and maintain. As a result, while the approach showed the potential of such an approach the existing language and compiler technology at the time was not sufficiently advanced for the needs of our RTRT work.

This led to the joint work between the graphics groups at Saarland University, DFKI, both headed by Prof. Slusallek, as well as the Compiler Construction group at Saarland University led by Prof. Sebastian Hack to design AnyDSL.

AnyDSL for the first time allows for creating different conceptional abstractions within a single language that are entirely “compiled away” for delivering high-performance implementations for different hardware architectures. This technology is currently being used to develop novel implementations of RTRT and lighting simulation algorithms that can easily be ported, adapted, and optimized for new or different hardware architectures. It aims at achieving optimal performance on different hardware architectures while enabling the formulation of algorithms at a very high level of abstraction and minimizing the efforts required for mapping it in an optimized way to efficient hardware-specific code.

So far, the first publications on AnyDSL describe some of the technologies involved in the design of AnyDSL [Danilewski 14] as well as its application to image processing [Köster 14]. The formulation of the abstractions for RTRT and global lighting simulation including their mapping to specific hardware is still ongoing.

6. Consequences for Dreamspace

Based on the above review of technologies developed over the last 15 years as well as the experience with related work from other researchers around the globe, we currently draw the following conclusions for the Dreamspace project:

In order to be able to develop algorithms that are highly efficient, portable across hardware architectures, and can easily be adapted and maintained, we plan to use the AnyDSL technology as the basis for our work.
Currently, we are simultaneously pushing the development of AnyDSL itself (the main focus in the first months of the project) and the development of conceptual abstractions for RTRT in AnyDSL (started recently). We expect these activities to continue for about 6 months during wich the focus will shift more towards RTRT development and less AnyDSL-specific work.
For our RTRT architecture, we aim at starting from the approaches currently implemented and optimized for Embree. We can thus leverage the considerable efforts that went into that system, while gaining additional flexibility and optimization opportunities through AnyDSL.
Regarding the global illumination strategy, we will start with a simple path tracer as its implementation is rather simple and very well understood. It is also the base-line approach supported by most related systems in the movie industry (e.g. the Arnold renderer[1]). Subsequently, we will aim at the integration of full VCM support [Georgiev 12b] based on our recent GPU implementations [Davidovic 14]. The goal is not to have full-fledged support for all possible rendering features but focus on a real-time performance with basic functionality. However, the system should be designed such that it is easy to add more advanced features.
While the AnyDSL development is still ongoing, we will make use of Embree as the key real-time renderer for the Dreamspace project. The current version has some limitations that restrict the features supported for pre-visualization. However, we are currently not considering extending its feature set but rather focus on replacing it with the AnyDSL-based implementation as early as possible.
In parallel to the above, activities to build a complete system are already ongoing. The system will be comprised of (i) a client architecture for the display and potentially editing of the scene by different people and from multiple perspectives, (ii) a real-time scene synchronization server that forwards and synchronizes scene changes between all components of the system, and (iii) a (cluster of) rendering servers that generate new views and streams them to the respective clients. A key goal here is to minimize the full round-trip latency through all components to be sufficient for high-quality pre-visualization purposes. An early prototype version of the full system is already being tested and evaluated using Embree. The new RTRT code base will be integrated as early as possible, once it starts generating first images.
Given the state and the performance of current algorithms and existing hardware architectures it is clear that considerable hardware resources will be needed to drive (multiple) clients at real-time rates and in low latency. This was clear from the start of the project. However, recently Intel officially announced their new Knights Landing processor architecture for 2H2015. This architecture seems very well suited for the goals of the Dreamspace project. Since there exist good connections to Intel through the partner IVCI, we suggest to focus on the development activities on this hardware architecture for now, as the Embree code base contains code that seems to be a good starting point and the architecture offers more performance and is more flexible than existing GPUs [Woop14].

7. Conclusions

The Dreamspace project has set itself very ambitious goals trying to provide real-time and low-latency remote rendering at photorealistic qualities with global illumination. However, given the algorithms and the advanced implementation tools as discussed above we can conclude that we have the means to achieve these goals.

It should be noted however, that they remain challenging goals and that we need to make good progress along all lines of development to be able to achieve them!

[1] http://www.solidangle.com/

8. List of Most Relevant Papers

The following list of papers contains the most relevant ones for the work in Dreamspace. For copyright reasons most of the paper contain links to their online versions.

1. [Georgiev 12b]: Light Transport Simulation with Vertex Connection and Merging
   ACM Transactions on Graphics (Proceedings Siggraph Asia), 2012
   https://graphics.cg.uni-saarland.de/2012/vertex-connection-and-merging/
2. [Georgiev 12a]: Importance Caching for Complex Illumination
   Computer Graphics Forum, 2012
   https://graphics.cg.uni-saarland.de/2012/importance-caching-for-complex-illumination/
3. [Davidovic 14]: Progressive Light Transport on the GPU: Survey and Improvements
   ACM Transactions on Graphics (TOG, Proceedings of Siggraph 2014), 2014
   Paper will become available online after Siggraph, August 2014, in the ACM Digital Library
4. [Woop 14]: Embree - A Kernel Framework for Efficient CPU Ray Tracing
   ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 2014
   Paper will become available online after Siggraph, August 2014, in the ACM Digital Library
5. [Wald 01a]: Interactive Rendering with Coherent Ray-Tracing
   Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001), 2001
   https://graphics.cg.uni-saarland.de/2001/wald2001ircrt2/
6. [Wald 02]: Interactive Global Illumination using Fast Ray Tracing
   13th EUROGRAPHICS Workshop on Rendering, 2002.
   https://graphics.cg.uni-saarland.de/2002/wald02igi/
7. [Benthin 03]: A Scalable Approach to Interactive Global Illumination
   Computer Graphics Forum, 2003
   https://graphics.cg.uni-saarland.de/2003/benthin03igi2/
8. [Günther 04]: Realtime Caustics using Distributed Photon Mapping
   Rendering Techniques (Proceedings of Eurographics Symposium on Rendering 2004), 2004
   https://graphics.cg.uni-saarland.de/2004/guenther04rtpm/
9. [Popov 06]: Experiences with Streaming Construction of SAH KD-Trees
   Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing, 2006
   https://graphics.cg.uni-saarland.de/2007/popov07gpurt/
10. [Woop 05]: RPU: A Programmable Ray Processing Unit for Realtime Ray Tracing
    ACM SIGGRAPH 2005
    https://graphics.cg.uni-saarland.de/2005/woop05rpu/
11. [Georgiev 08]: RTfact: Generic Concepts for Flexible and High Performance Ray Tracing
    IEEE/EG Symposium on Interactive Ray Tracing 2008
    https://graphics.cg.uni-saarland.de/2008/georgiev08rtfact/
12. [Köster 14]: Platform-Specific Optimization and Mapping of Stencil Codes through Refinement
    Proceedings of HiStencil 2014
    https://graphics.cg.uni-saarland.de/2014/platform-specific-optimization-and-mapping-of-stencil-codes-through-refinement/
13. [Karrenberg 10]: AnySL: Efficient and Portable Shading for Ray Tracing
    Proceedings of the Conference on High Performance Graphics, 2010
    https://graphics.cg.uni-saarland.de/2010/karrenberghpg2010/

9. References

Publications marked in red below have been selected as most relevant for the work in Dreamspace and a link to the online version is given above.

[Appel 68] A. Appel. *Some techniques for shading machine renderings of solids*. **SJCC**, pages 27–45, 1968.

[Benthin 03] Benthin, Carsten, Wald, Ingo and Slusallek, Philipp, *A Scalable Approach to Interactive Global Illumination*, **Computer Graphics Forum (Proceedings of Eurographics 2003)**, 22(3):621—630, 2003.

[Benthin 04] Benthin, Carsten, Wald, Ingo and Slusallek, Philipp, *Interactive Ray Tracing of Free-Form Surfaces,* Proceedings of **Afrigraph 2004**, 2004.

[Benthin 06a] Benthin, Carsten, Wald, Ingo and Slusallek, Philipp, *Techniques for Interactive Ray Tracing of Bèzier Surfaces,* **Journal of Graphics Tools**, 11(2):1-16, 2006.

[Benthin 06b] Benthin, Carsten, Wald, Ingo, Scherbaum, Michael and Friedrich, Heiko, *Ray Tracing on the CELL Processor,* Proceedings of the **2006 IEEE Symposium on Interactive Ray Tracing**, page 15—23, 2006.

[Cook 84] Robert L. Cook, Thomas Porter, Loren Carpenter, *Distributed Ray Tracing,* Proceedings of **ACM Siggraph 1984**, 18(3), 137-145, 1984.

[Danilewski 14] Piotr Danilewski, Marcel Köster, Roland Leißa, Richard Membarth, Philipp Slusallek, *Specialization through Dynamic Staging, a*ccepted for publication at **13th International Conference on Generative Programming: Concepts & Experiences (GPCE'14)**, 2014.

[Davidovic 10] Davidovic, Tomas, Krivanek, Jaroslav, Hasan, Milos, Slusallek, Philipp and Bala, Kavita, Combining global and local virtual lights for detailed glossy illumination, **ACM Transactions on Graphics**, 29:143:1--143:8, 2010.

[Davidovic 11] Davidovic, Tomas, Marsalek, Lukas and Slusallek, Philipp, *Performance Considerations When Using a Dedicated Ray Traversal Engine*, 19th **International Conference on Computer Graphics, Visualization and Computer Vision 2011 (WSCG 2011)**, Pilsen, page 65—72, 2011.

[Davidovic 14] Tomas Davidovic, Jaroslav Krivanek, Milos Hasan, Philipp Slusallek, Progressive Light Transport on the GPU: Survey and Improvements, in **ACM Transactions on Graphics (TOG)**, accepted for publication, 2014.

[Dietrich 03] Dietrich, Andreas, Wald, Ingo, Benthin, Carsten and Slusallek, Philipp, *The OpenRT Application Programming Interface - Towards A Common API for Interactive Ray Tracing,* Proceedings of the **2003 OpenSG Symposium**, page 23—31, 2003.

[Dietrich 04] Dietrich, Andreas, Wald, Ingo, Wagner, Markus and Slusallek, Philipp, *VRML Scene Graphs on an Interactive Ray Tracing Engine,* **Proceedings of IEEE VR 2004**, page 109—116, 2004.

[Dietrich 05] Dietrich, Andreas, Colditz, Carsten, Deussen, Oliver and Slusallek, Philipp, Realistic and Interactive Visualization of High-Density Plant Ecosystems, Natural Phenomena 2005, Proceedings of the **Eurographics Workshop on Natural Phenomena**, page 73—81, 2005.

[Dietrich 06] Dietrich, Andreas, Marmitt, Gerd and Slusallek, Philipp, *Terrain Guided Multi-Level Instancing of Highly Complex Plant Populations,* Proceedings of the **2006 IEEE Symposium on Interactive Ray Tracing**, page 169—176, 2006.

[Dietrich 06b] Dietrich, Andreas, Wald, Ingo, Schmidt, Holger, Sons, Kristian and Slusallek, Philipp, *Realtime Ray Tracing for Advanced Visualization in the Aerospace Industry,* Proceedings of the 5th Paderborner **Workshop Augmented & Virtual Reality in der Produktentstehung**, 2006.

[Dietrich 07a] Dietrich, Andreas, Stephens, Abe and Wald, Ingo, *Exploring a Boeing 777: Ray Tracing Large-Scale CAD Data,* **IEEE Computer Graphics and Applications**, 27(6):36—46, 2007.

[Dietrich 07b] Dietrich, Andreas, Gobbetti, Enrico and Yoon, Sung-Eui, *Massive Model Rendering Techniques,* **IEEE Computer Graphics and Applications**, 27(6):20—34, 2007.

[Dietrich 07c] Dietrich, Andreas, Marmitt, Gerd and Slusallek, Philipp, *Trillion Triangle Terrain,* **Computer Graphics Forum**, 26(1):129—130, 2007.

[Georgiev 08] Georgiev, Iliyan and Slusallek, Philipp, *RTfact: Generic Concepts for Flexible and High Performance Ray Tracing,* Proceedings of the **IEEE/EG Symposium on Interactive Ray Tracing 2008**, page pp. 115-122, 2008.

[Georgiev 10] Georgiev, Iliyan and Slusallek, Philipp, *Simple and Robust Iterative Importance Sampling of Virtual Point Lights,* Proceedings of **Eurographics 2010**, 2010.

[Georgiev 12a] Georgiev, Iliyan, Krivanek, Jaroslav, Popov, Stefan and Slusallek, Philipp, *Importance Caching for Complex Illumination,* **Computer Graphics Forum**, 31(2), 2012.

[Georgiev 12b] Georgiev, Iliyan, Krivanek, Jaroslav, Davidovic, Tomas and Slusallek, Philipp, *Light Transport Simulation with Vertex Connection and Merging,* **ACM Transactions on Graphics (Proceedings of Siggraph Asia 2012)**, 192:1--192:10, 2012.

[Georgiev 13] Iliyan Georgiev, Jaroslav Krivanek, Toshiya Hachisuka, Derek Nowrouzezahrai, Wojciech Jarosz, *Joint Importance Sampling of Low-Order Volumetric Scattering, in* **ACM Transactions on Graphics (Proceedings Siggraph Asia 2013)**, 32, 6, 2013.

[Günther 04] Günther, Johannes, Wald, Ingo and Slusallek, Philipp, *Realtime Caustics using Distributed Photon Mapping,* Rendering Techniques 2004, Proceedings of the **Eurographics Symposium on Rendering**, page 111—121, 2004.

[Günther 06a] Günther, Johannes, Friedrich, Heiko, Wald, Ingo, Seidel, Hans-Peter and Slusallek, Philipp, Ray Tracing Animated Scenes using Motion Decomposition, **Computer Graphics Forum (Proceedings of Eurographics 2006)**, 25(3):517—525, 2006.

[Günther 06b] Günther, Johannes, Friedrich, Heiko, Seidel, Hans-Peter and Slusallek, Philipp, *Interactive Ray Tracing of Skinned Animations,* **The Visual Computer**, 22(9):785—792, 2006.

[Günther 07] Günther, Johannes, Popov, Stefan, Seidel, Hans-Peter and Slusallek, Philipp, *Realtime Ray Tracing on GPU with BVH-based Packet Traversal,* Proceedings of the **IEEE/Eurographics Symposium on Interactive Ray Tracing 2007**, page 113—118, 2007.

[Jensen 95] Henrik Wann Jensen and Niels Jørgen Christensen, *Photon Maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects*, **Computers & Graphics**, vol. 19 (2), pages 215-224, 1995.

[Jensen 01] Henrik Wann Jensen, *Realistic Image Synthesis using Photon Mapping*, ISBN: 1-56881-140-7. AK Peters, 2001

[Kajiya 86] Kajiya, James, *The rendering equation*, **Proceedings of Siggraph 1986**, 143, 1986.

[Kalojanov 09] Kalojanov, Javor and Slusallek, Philipp, *A Parallel Algorithm for Construction of Uniform Grids,* Proceedings of **High Performance Graphics 2009**, 2009.

[Kalojanov 11] Kalojanov, Javor, Billeter, Markus and Slusallek, Philipp, *Two-Level Grids for Ray Tracing on GPUs,* **Computer Graphics Forum**, 2011.

[Karrenberg 10] Karrenberg, Ralf, Rubinstein, Dmitri, Slusallek, Philipp and Hack, Sebastian, *AnySL: Efficient and Portable Shading for Ray Tracing,* Proceedings of the **Conference on High Performance Graphics**, 2010.

[Köster 14] Marcel Köster, Roland Leißa, and Sebastian Hack, Richard Membarth and Philipp Slusallek, *Platform-Specific Optimization and Mapping of Stencil Codes through Refinement,* in **Proceedings of HiStencil 2014**, 2014.

[Löffler 11] Löffler, Alexander, Marsalek, Lukas, Hoffmann, Hilko and Slusallek, Philipp, Realistic Lighting Simulation for Interactive VR Applications, Virtual Environments 2011: Proceedings of **Joint Virtual Reality Conference of euroVR and EGVE (JVRC)**, page 1—8, 2011.

[Marmitt 04] Marmitt, Gerd, Kleer, Andreas, Wald, Ingo, Friedrich, Heiko and Slusallek, Philipp, *Fast and Accurate Ray-Voxel Intersection Techniques for Iso-Surface Ray Tracing,* **Vision, Modelling, and Visualization 2004 (VMV), Stanford (CA)**, USA, 2004.

[Marsalek 10] Marsalek, Lukas, Georgiev, Iliyan, Dehof, Anna Katharina, Lenhof, Hans-Peter, Slusallek, Philipp and Hildebrandt, Andreas, *Real-Time Ray Tracing of Complex Molecular Scenes,* **14th International Conference on Information Visualisation** (IV), page 239—245, 2010.

[Parker 98] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen, and Peter Pike Sloan. *Interactive ray tracing for isosurface rendering*. In **IEEE Visualization '98**, pages 233–238, 1998.

[Parker 10] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, Martin Stich, *OptiX: A General Purpose Ray Tracing Engine*, in **ACM Transactions on Graphics (SIGGRAPH 2010 Proceedings)**, August 2010

[Popov 06] Popov, Stefan, Günther, Johannes, Seidel, Hans-Peter and Slusallek, Philipp, *Experiences with Streaming Construction of SAH KD-Trees,* Proceedings of the **2006 IEEE Symposium on Interactive Ray Tracing**, page 89—94, 2006.

[Popov 07a] Popov, Stefan, Günther, Johannes, Seidel, Hans-Peter and Slusallek, Philipp, *Stackless KD-Tree Traversal for High Performance GPU Ray Tracing*, **Computer Graphics Forum**, 26(3), 2007.

[Popov 09] Popov, Stefan, Georgiev, Iliyan, Dimov, Rossen and Slusallek, Philipp, *Object Partitioning Considered Harmful: Space Subdivision for BVHs,* HPG '09: Proceedings of the **1st ACM conference on High Performance Graphics**, page 15—22, 2009.

[Popov 13] Popov, Stefan, Georgiev, Iliyan, Slusallek, Philipp and Dachsbacher, Carsten, *Adaptive Quantization Visibility Caching*, **Computer Graphics Forum**, 32(2), 2013.

[Repplinger 09] Repplinger, Michael, Löffler, Alexander, Thielen, Martin and Slusallek, Philipp, *A Flexible Adaptation Service for Distributed Rendering,* Proceedings of **9th Eurographics Symposium on Parallel Graphics and Visualization 2009 (EGPGV '09)** , page 49—56, 2009.

[Rubinstein 09] Rubinstein, Dmitri, Georgiev, Iliyan, Schug, Benjamin and Slusallek, Philipp, *RTSG: Ray Tracing for X3D via a Flexible Rendering Framework,* Proceedings of the **14th International Conference on 3D Web Technology 2009 (Web3D Symposium '09)** , page 43—50, 2009.

[Schmittler 02] Schmittler, Jörg, Wald, Ingo and Slusallek, Philipp, *SaarCOR - A Hardware Architecture For Ray Tracing,* Proceedings of the **Conference on Graphics Hardware 2002**, page 27—36, 2002.

[Schmittler 03] Schmittler, Jörg, Leidinger, Alexander and Slusallek, Philipp, *A Virtual Memory Architecture for Real-Time Ray Tracing Hardware,* **Computer & Graphics**, page 0097—8493, 2003.

[Schmittler 04] Schmittler, Jörg, Woop, Sven, Wagner, Daniel, J. Paul, Wolfgang and Slusallek, Philipp, *Realtime Ray Tracing of Dynamic Scenes on an FPGA Chip,* Proceedings of **Graphics Hardware**, page 95—106, 2004.

[Stich 09] Stich, Martin, Friedrich, Heiko and Dietrich, Andreas, *Spatial Splits in Bounding Volume Hierarchies,* Proceeding of **High Performance Graphics**, 2009.

[Veach 97] Eric Veach and Leonidas J. Guibas, *Metropolis Light Transport,* in Proceedings of **SIGGRAPH 1997**, pp. 65-76, 1997.

[Wald 01a] Ingo Wald, Carsten Benthin, Markus Wagner, and Philipp Slusallek, *Interactive Rendering with Coherent Ray-Tracing*, in **Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001)**, pp 153-164, 20(3), 2001

[Wald 01b] Wald, Ingo, Slusallek, Philipp and Benthin, Carsten, *Interactive Distributed Ray Tracing of Highly Complex Models,* In S.J.Gortler and K.Myszkowski, editor, **Rendering Techniques 2001 (Proceedings of the 12th EUROGRAPHICS Workshop on Rendering)**, page 277—288, 2001.

[Wald 02] Wald, Ingo, Kollig, Thomas, Benthin, Carsten, Keller, Alexander and Slusallek, Philipp, *Interactive Global Illumination using Fast Ray Tracing,* Proceedings of the **13th EUROGRAPHICS Workshop on Rendering**, 2002.

[Wald 03a] Wald, Ingo, Benthin, Carsten and Slusallek, Philipp, *Interactive Global Illumination in Complex and Highly Occluded Environments,* Proceedings of the **14th Eurographics Workshop on Rendering**, 2003.

[Wald 03b] Wald, Ingo, J. Purcell, Timothy, Schmittler, Joerg, Benthin, Carsten and Slusallek, Philipp, **Eurographics State of the Art Reports**, 2003.

[Wald 03c] Wald, Ingo, Benthin, Carsten and Slusallek, Philipp, *Distributed Interactive Ray Tracing of Dynamic Scenes,* Proceedings of the **IEEE Symposium on Parallel and Large-Data Visualization and Graphics (PVG)**, 2003.

[Wald 04a] Wald, Ingo, Dietrich, Andreas and Slusallek, Philipp, *An Interactive Out-of-Core Rendering Framework for Visualizing Massively Complex Models,* **Rendering Techniques 2004 (Proceedings of the Eurographics Symposium on Rendering)**, page 81—92, 2004.

[Wald 04b] Wald, Ingo, Günther, Johannes and Slusallek, Philipp, *Balancing Considered Harmful -- Faster Photon Mapping using the Voxel Volume Heuristic,* **Computer Graphics Forum** Volume 22, 2004.

[Wald 05a] Wald, Ingo, Friedrich, Heiko, Marmitt, Gerd, Slusallek, Philipp and Seidel, Hans-Peter, *Faster Isosurface Ray Tracing Using Implicit KD-Trees*, **IEEE Transactions on Visualization and Computer Graphics**, 11(5):562-572, 2005.

[Wald 05b] Ingo Wald and Hans-Peter Seidel, *Interactive Ray Tracing of Point-based Models,* Proceedings of the **Symposium on Point Based Graphics**, 2005.

[Whitted 80] T. Whitted. *An improved illumination model for shaded display*., 23(6):343–349, June 1980.

[Woop 05] Woop, Sven, Schmittler, Jörg and Slusallek, Philipp, *RPU: A Programmable Ray Processing Unit for Realtime Ray Tracing,* Proceedings of **ACM SIGGRAPH 2005**, 2005.

[Woop 06a] Woop, Sven, Marmitt, Gerd and Slusallek, Philipp, *B-KD Trees for Hardware Accelerated Ray Tracing of Dynamic Scenes,* Proceedings of **Graphics Hardware**, 2006.

[Woop 06b] Sven Woop, Erik Brunvand and Slusallek, Philipp, *Estimating Performance of a Ray-Tracing ASIC Design,* Proceedings of **IEEE Symposium on Interactive Ray Tracing 2006**, page 7-14, 2006.

[Woop 14] Ingo Wald, Sven Woop, Carsten Benthin, Gregory S Johnson, and Manfred Ernst, *Embree - A Kernel Framework for Efficient CPU Ray Tracing,* **ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 2014**, accepted for publication.