Eurographics Workshop on Natural Phenomena (2005)
E. Galin, P. Poulin (Editors)

Realistic and Interactive Visualization of High-Density Plant Ecosystems

Andreas Dietrich[†], Carsten Colditz[‡], Oliver Deussen[‡], and Philipp Slusallek[†]

[†]Computer Graphics Group,
Saarland University, Saarbrücken
{dietrich,slusallek}@cs.uni-sb.de

[‡]Computer Graphics & Media Design,
University of Konstanz
{colditz,deussen}@inf.uni-konstanz.de

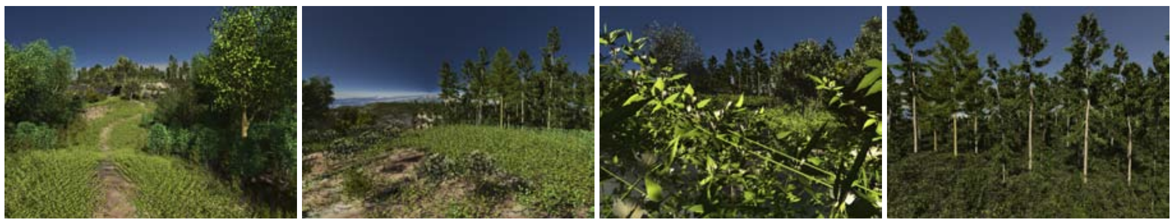
Figure 1: *Example screenshots taken with our framework during a walkthrough of a complex plant ecosystem model containing more than 365,000 individual plants. Note the extremely high geoemetric detail, resulting in a total of over 1.5 billion polygons.*

Abstract

Modeling and visualization of natural and realistic outdoor ecosystems like forests or meadows remains a highly challenging task in computer graphics. Convincing and non-artificial scenes require many thousands to millions of plants, each modeled with a high degree of geometric, shading, and lighting detail. At the same time many applications demand interactive rendering at high fidelity.
In this paper we demonstrate a ray tracing based approach that does not rely on geometric simplification. Using real-time ray tracing technology, we are able to directly render the highly complex original models with advanced shading and lighting, including pixel-accurate shadows, and even high dynamic range lighting from environment maps. On high-end PCs we obtain interactive performance with reduced quality that quickly converges to high quality in static situations. On clusters of PCs high quality can be sustained even at interactive rates. We demonstrate these results using a highly complex forest model.

Categories and Subject Descriptors (according to ACM CCS): I.3.2 [Computer Graphics]: Distributed/network graphics I.3.7 [Computer Graphics]: Ray tracing I.6.2 [Simulation and Modeling]: Applications

1. Introduction

Realistic and faithful display of large natural plant ecosystems like e.g., forests, meadows, gardens, or parks, is a highly demanding task in computer graphics. An accurate model representation of such environments often contains complex vegetation, often consisting of many hundreds of thousands of individual plants from dozens of different species. All plants need to be modeled with a high degree of detail, resulting in scenes consisting of billions of geometric primitives with complex lighting and shading.

The ability for interactive walkthroughs and/or to quickly render still images of such outdoor scenes have important applications e.g. in architectural planning processes, garden and landscape design, but also in computer animations and games. Goals for our research have been to significantly improve the achievable realism for outdoor environments while still allowing for interactive or close to interactive performance depending on the available hardware. In particular these goals include:

- Support for highly realistic and detailed plant models.
- Realistic surface shading using detailed textures and reflection models.
- Realistic lighting including shadows and illumination from High Dynamic Range (HDR) environment data.
- Support for arbitrary movements of the viewer (e.g. from overview of a meadow to the close-up of a single leaf), while minimizing rendering artifacts and maintaining good interactive performance.

Although global illumination is a significant factor in outdoor environments, we have not attempted to account for it in this paper.

While a number of techniques for interactive rendering of natural scenes have been developed (see Section 2), they are often tailored for specific applications or limited environments. Interactive handling of complex scene databases in combination with advanced shading and lighting is usually not possible.

1.1. Ray Tracing of Natural Scenes

When it comes to physically-correct illumination and photorealistic visualization, ray tracing algorithms [Gla89] are usually the premier choice. Due to the fact that they can accurately model the physics of light transport and have logarithmic time complexity regarding scene complexity, they are particularly suited for rendering of large natural scenes.

Ray tracing can easily incorporate different shading and lighting models in a plug-and-play manner. This enables a convenient combination of a variety of natural phenomena effects such as e.g., transparent plant leafs, atmospheric effects like volumetric clouds, environmental lighting depending on different sky models, and other effects.

Individually modeling every single plant in highly complex environments would require huge amounts of memory for representing the geometry alone. Instead, one usually relies on numerous and flexible instantiations of a few individual but highly detailed plants. Ray tracing allows to very efficiently handle such instances where they become visible (see Section 4).

However, in the past ray tracing had been notoriously slow and had therefore been exclusive used for offline image generation. With the recent progress in interactive and real-time ray tracing [Wal04], advanced visualization of large models has finally become possible. Nonetheless, interactive photorealistic lighting and rendering of *complex* outdoor environments still poses a significant challenge.

In this paper we demonstrate first steps into this direction, targeting interactive walkthrough of complex landscapes, containing over 365,000 single plants with a total of more than 1.5 billion primitives, which are rendered without any geometric simplification. The scene can be displayed even with pixel-accurate shadows, transparent leaves, soft shadows, and high dynamic range lighting via environment maps. Running on a cluster of standard PCs, the presented system allows for easy trade-off between image quality and interactive performance. Additionally, it features a progressive mode, where the image is continuously refined during no user interaction, which results in high-resolution, anti-aliased images with complex lighting within a few seconds.

In the next section we provide a short overview over existing techniques and systems for rendering large natural environments. Section 3 will then describe the synthetic generation of natural ecosystems, followed by a presentation of our ray tracing techniques in Section 4. Section 5 presents results of applying the ray tracing system to large outdoor scenes. We conclude in Section 6 and outline future research activities in Section 7.

2. Related Work

The Problem of rendering complex natural ecosystems has already received a lot of attention. We will first briefly give an overview over common techniques mainly targeted at rasterization-based image generation, followed by some background about ray tracing being applied to massively complex scenes.

2.1. Level of Detail

An earlier work that explicitly dealt with the problem of detail reduction of tree models was presented by Rossignac and Borrel [RB93]. They proposed a method applicable to such data that determines groups of nearby vertices, and collapses them to a single vertex without being concerned with the topology. The method works, but the appearance of the models changes since normals are altered.

As early as 1985, Reeves and Blau [RB85] represented trees using collections of disks in order to reduce the complexity of the foliage. Carefully selecting colors and shadowing effects resulted in very esthetic images though the biological correctness of the models was very poor. Weber and Penn [WP95] proposed a method of approximating trees by points and lines. The foliage was represented by a set of points that were placed inside agglomerated leaves of the foliage, which they referred to as “masses”. Lines were used for the tree skeleton. The idea of using point clouds for representing trees and other objects was further developed by several authors [SD01,DCSD02,DVS03]. A similar idea was presented by Wand et al. [WFP*01]. For each triangle of a scene it was determined whether it was shown polygonally or point wise. Very efficient computing methods allowed to display scenes of extreme complexity.

Shade et al. [SGHS98] used Layered Depth Images (LDI) to render complex objects from pre-computed pixel-based representations with depth at places where full polygonal representations are too expensive and impostors such as billboards are too inaccurate. Chang et al. [CBL99] developed a hierarchical form of LDI that allowed them to generate different levels of detail. Both approaches are pixel-based, which results in low image quality for close-ups. Max [Max96] modeled and rendered trees hierarchically from pre-computed images with Z-buffers. A twig consists of some images of leaves, a branch of several images of twigs. Viewed from a distance, the whole tree is represented by one single image, if the viewer comes closer, the image is replaced by a combination of the first order branch images. A finer representation is achieved by replacing the images of first order branches by several twig images and so on. Though Max et al. [MDK99] accelerate the process by using texture hardware, the rendering times are far above of what is needed for interactive applications due to extensive texture transfer operations.

To crossfade smoothly between a geometry representation and the corresponding billboard representation of the model, in [CCDH05] a dynamic transition is used. At close range, only the model with full geometry complexity will be displayed. In the distance only the billboards will be rendered. Between these two static states the representation will be changed dynamically. The mode of operation is simple and guarantees good visual results without the need of complex blending operations. With increasing distance, the geometry of the original model will be reduced and in the inverse way, the value of the alpha-channel test for the billboard rendering process will be changed. Consequently, the billboard becomes more visible with every step.

2.2. Billboard Representation

In contrast to level of detail techniques, billboards have been used for the representation of tree models for many years, see e.g., [RH94]. In its simplest form a billboard represents a whole model, and therefore it can only be used in the background since no parallax distortion can be handled. A better approximation can be achieved using volumetric representations. Jakulin uses sets of parallel billboards [Jak00] to represent trees, however, due to the parallel orientation artifacts occur.

Neyret [Ney98] converts complex natural objects into volumetric textures, which are then ray traced. MIP maps are used to reduce texture data and to generate samples from distant trees. The method provides high-quality images of very complex scenes at a moderate rendering time of several minutes, but it is not applicable to interactive purposes. Recently, this technique was adapted to graphics hardware by the authors [DN04], however, an efficient level of detail mechanism on the billboards is necessary to cope with the enormous amount of textures that are needed to represent large scenes.

In [BCF*05] a similar technique is used to render larger parts of scenes in the background, and a method for achieving level of detail on volumetric textures is presented. To represent tree models using billboards, the "billboard clouds" approach can be used [DDSD03]. It represents a geometry model by a set of arbitrarily oriented billboards. In this work an automatic method for finding a good set is presented, however, for tree models the proposed algorithm is not an optimal solution, since the used plane-space transform is not able to provide significant results due to the non-compact geometry of our tree models. [BCF*05] presents a clustering algorithm on the basis of the model vertices that allows to find good, hierarchical billboard approximations.

2.3. Ray Tracing

Ray tracing of plant scenes has been done for over twenty years now, early examples include [KK86] and [SB87]. In order to handle scenes of a more realistic complexity Pharr et al. [PKGH97] have proposed a caching and reordering mechanism that reorders the rays in a way that minimizes disk I/O. Though this allows for efficiently ray tracing datasets much larger than main memory, it is difficult to employ in an interactive context. A system specifically designed for high-quality rendering of natural ecosystems presented by Deussen et al. [DHL*98] made use of a number of different techniques like approximated instancing, sub-scene compositing, and memory-coherent ray tracing. The system could render models with thousands of plants, build out of several million polygons, however, it has been exclusively used for offline rendering.

Interactive ray tracing has so far mainly found its application in visualization of highly complex CAD databases. The OpenRT real-time ray tracing engine [Wal04, WBDS03] has been shown to be capable of processing scenes up to several million triangles in real-time. Additionally, it incorporates physically correct and global lighting simulation, and features interactive placement of geometric parts. Wald et al. [WDS04] also presented out-of-core rendering variants that made it possible to interactively visualize a complete Boeing 777 CAD dataset containing more than 350 million individual and uninstantiated surface polygons.

3. Modeling of Realistic High-Density Ecosystems

The modeling of complex outdoor scenes is a challenge itself. The first step is to create a model library that contains the necessary species. There is a number of representations for creating plant models such as L-systems [PL90, Pru], plants based on growth models such as natFX [nat], or, in our case, Xfrog plant models [DL05]. These plant models are then combined into complete scenes using GIS data or manually edited using special editors where plant positions are handled similarly to paint programs [DHL*98]. The resulting data is a scene description consisting of plant positions and additional data such as species and/or model description. However, since for very large scenes even the storage of the plant positions is prohibitive due to the huge amount of data, functional specifications must be created that are converted to positions only for those parts of the scene that are shown. As stated in the introduction, our test scene consists of 365,000 plant instances and can therefore be handled by an interactive interface that works on explicit plant positions. The user is able to insert, delete, or move plant instances and to alter their position due to biological interaction processes. Figure 2 shows an interactive editing session.

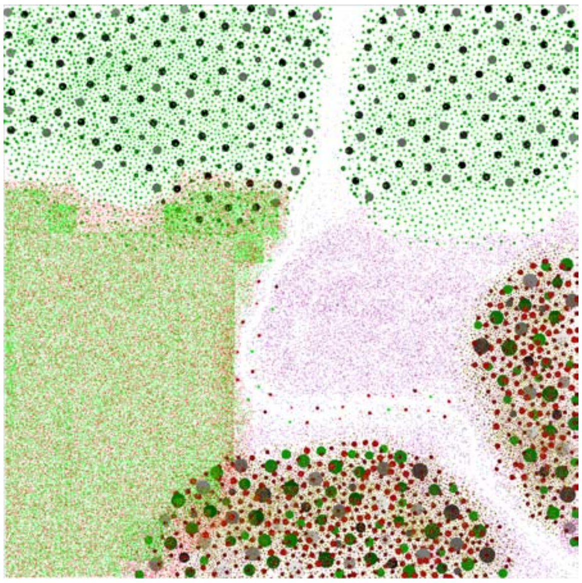
Figure 2: *Description of the scene in the interactive editor. The colors denote the different species, the size of the circles their relative size.*

Most of our scenes are divided into up to several layers that represent different types of plants such as ground-covering plants, shrubs, or trees. In our test scene we used six layers that are shown in Table 1.

This table also shows one of the problems we face in such scenes: Most of the geometric effort is spent for the smaller plants (layer 4), and, as the images show, this is still not enough and should be represented by a higher number of objects. In [BCF*05] a possible solution on the basis of shell textures (sets of parallel semi transparent textures) is presented that, however, is not so well suited for ray tracing. In the current version, these plants are shown as ordinary triangular models.

Based on the above description the scene is divided spatially in order to enable the rapid ray tracing. This is described in the next section.

4. Ray Tracing Highly-Detailed Natural Environments

Our framework builds on the OpenRT ray tracing core [Wal04]. The OpenRT engine has already been shown to be able to cope with large outdoor environments, which has been demonstrated, for example, with a field consisting of 28,000 sunflowers [WBDS03]. However, the structure and topology of that scene is very simple since it contains only a very limited number of plant species, and only two vegetation layers.

Realistically structured scenes, like the one we are considering in this paper, are composed of far more plants and vegetation layers, also they are highly unregular (see Section 3). As a result the system had to be changed to accommodate the complex scene and ensure a realistic representation while preserving interactivity.

4.1. Hierarchy of Spatial Indexes

In order to reduce unnecessary ray-triangle intersection tests the OpenRT system employs a two-level spatial kd-tree index [Ben75, WBS03]. Here, separate geometric objects maintain their own index, while the bounding boxes of the objects are themselves organized in a top-level kd-tree.

This setup has two primary functions: First, it allows for efficient object instantiation by simply keeping references to the geometric objects in conjunction with corresponding transformation matrices. During rendering, rays that hit an object in the top-level index are transformed into object space, where they are traversed and tested against the object's triangles. Second, it enables dynamic or animated scenes with rigid-body motion of objects. In this case only the top-level index has to be rebuilt if an object moves, because their internal kd-tree can be fully reused.

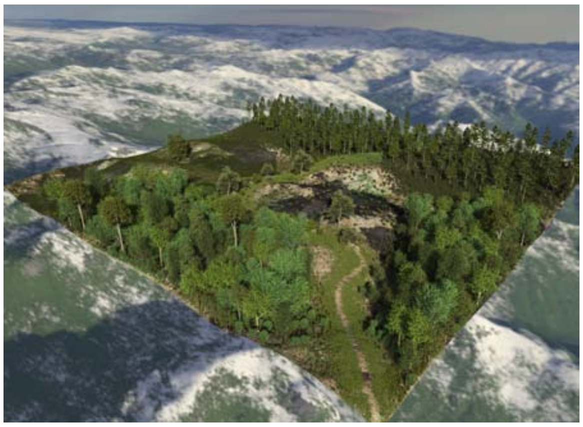
Figure 3: *Overview of the complete scene. Over 365,000 instances of 68 plants have been positioned on top of a 300m×300m polygonal height field.*

Layer	Species	Instances	average Complexity	total Triangles
layer 3 (higher haulm)	6	204,750	1,000	204,750,000
layer 4 (lower shrub)	19	144,091	5,000	720,455,000
layer 5 (higher shrub)	12	10,464	25,000	261,600,000
layer 6 (lower tree)	10	5,209	50,000	260,450,000
layer 7 (higher tree)	9	290	75,000	21,750,000
layer 8 (overlap tree)	12	237	100,000	23,700,000
Total	68	365.041		≈1.500.000.000

Table 1: *Different types of plants and their statistics in the test scene.*

Our test scene includes 68 individual plant models, which can be completely stored in main memory, including all object-level acceleration structures. More than 365,000 instances of these reference plants are then placed all over an 2 million polygon height field (see Figure 3).

To efficiently support dynamic scenes, the construction procedure of the top-level kd-tree had been designed to allow for fast rebuilding rather than the generation of highly optimized indexing structures. As a consequence, it only handles axis-aligned bounding boxes of the transformed objects. Due to the high density of plants in our test scene and because of their highly irregular positions, these bounding boxes often show a significant amount of overlap. This reduces the efficiency of the top-level index as all non-separable objects must be traversed until an intersection in this region can be reported as final.

This problem could be alleviated by replacing the fast top-level kd-tree construction algorithm with the more accurate but slower construction algorithm that is used for the rigid-body objects of the scene. In this approach cost prediction functions help to place kd-tree space splitting planes. As cost prediction function we used the so-called Surface Area Heuristic (SAH) [MB89]. While this increases the time needed to construct the top-level kd-tree, it decreases the per-frame rendering time, which is acceptable if only walk-throughs through a static world are intended as in our case.

4.2. Adaptive Transparency Control

Although our plant models are composed of up to hundreds of thousands of polygons, their leafs only consist of coarse triangle meshes. In addition to a diffuse color, each leaf texture incorporates an alpha-channel that is used to cut out the leaf's shape. Because this transparency is evaluated in a shader, we report an intersection, which stops traversal and triggers a call to the shader. The shader then evaluates the transparency and may continue tracing by spawning a new transparency ray.

This approach leads to a great number of secondary rays, due the highly complex scene geometry, the extensive use of leaf textures, and the extremely high density of objects (see Figure 4). In some cases more the 30 ray levels of recursion are necessary until a non-transparent surface is hit. Note, that the same problem applies to all other types of rays, e.g. shadow rays.

Because this high amount of transparency already occurs for close-by objects, it does not help to limit the maximum recursion depth, as this would drastically change a plant's appearance. We therefore use an adaptive scheme where the maximum recursion depth of secondary rays is continually decreased with the hit point's distance to the viewer. That way traversal can be aborted, i.e. the shader can decide not to fire a transparency ray, even if the alpha-channel signals transparency at the respective hit point. The diffuse color is then replaced by the averaged color of all non-transparent texels of the surface texture.

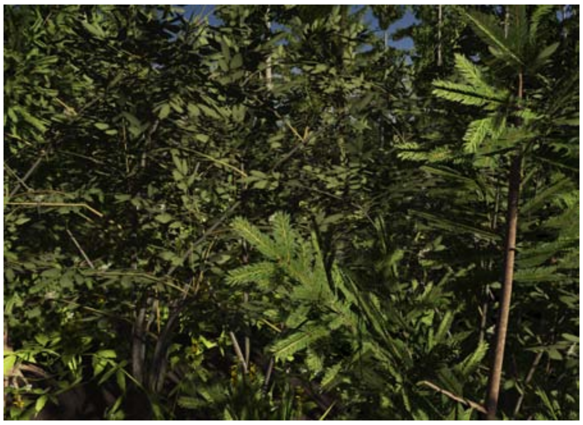

Figure 4: *Close-up view of some plants revealing their highly complex structure. Every leaf texture contains an alpha-channel that controls the leaf's geometric shape. Due to the high plant density this can lead to over 30 levels of recursion until a ray hits a non-transparent surface.*

While this still alters the appearance of distant objects, the visual impact is not that dramatic. Because of strong aliasing artifacts that stem from the vast number of small leafs, we have to apply pixel oversampling. Due to the high color variance for distant objects (especially if environmental lighting is used, see next Section), limited leaf transparency is not significantly noticeable when averaging colors that result from tracing multiple rays per pixel.

Note that techniques like level of detail are difficult to apply in this situation. Even if an object contains few polygons, the high object count and plant density would still lead to a large number of polygons covered by a single pixel.

4.3. Realistic Environmental Lighting

In addition to a high degree of detail, a realistic representation of natural outdoor scenes heavily depends on the incident illumination, i.e. the light coming from the environment. Approximation of natural lighting with e.g. a single directional light, is usually not sufficient (see Figure 5) as it cannot capture the subtle but important effects like soft shadows.

The cost of pre-calculating light transport, e.g. using methods from Precomputed Radiance Transfer (PRT) [SKS02], is not affordable in scenes of such a high complexity, in particular when making heavy use of instantiation. We therefore directly sample the incident radiance emanating from a high dynamic range environment map. In order to avoid noise, resulting from random sampling of such a map, we use a large but fixed number of virtual directional light sources in the spirit of [ARBJ03, KK03] that act as an approximation of the environment illumination. Unfortunately, it is too costly during shading to fire as many shadow rays as there are virtual light sources. For that reason we are picking only a small number of random samples from the full set of virtual directional lights.

Note that although the addition of environmental illumination provides significantly more realism, we still cannot afford to interactively compute a full global illumination solution for such scenes. Because of this, important effects like diffuse leaf transparency, as well as inter-object light scattering are currently not accounted for.

4.4. Interleaved Sampling

It has already mentioned that aliasing poses a difficult problem in scenes with the geometric complexity of our test scene. This does not only account for primary rays but for shadow rays, i.e. rays that sample the environment. To make best use of a given ray budget, it is advantageous to combine anti-aliasing due to geometry and illumination [KH01, BWS03]. Rather than using the same set of virtual directional lights for every primary ray, we can break up each set into subsets, each of which applies to a different

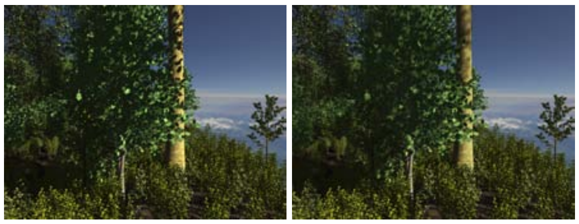

Figure 5: *Visual impact of environmental high dynamic range lighting. a) Approximate sun light using a single directional light source. b) Illumination based on an HDR environment map. Note particularly the soft shadows on the tree trunk and the tree foliage.*

set of primary rays. That way we can increase the number of light samples while simultaneously improving pixel-anti-aliasing.

4.5. Progressive Refinement

The above techniques allow for quickly generating high-quality images at interactive rates on a set of PCs. However, this visualization still requires significant compute power (see Section 5). In particular high-resolution images are quite demanding. As a result, our system can be operated in a special progressive mode: As soon as camera motion stops, we perform progressive improvement of image quality by computing successive images of the same viewpoint with new random samples, and accumulate the resulting images. This makes it possible to generate high-quality images of high-resolution even on a single PC in just a couple of seconds. During user interaction pictures with reduced quality can still give a very good impression of the final result (see Figure 6).

5. Results

All our experiments were conducted on a cluster of standard PCs. Each PC was equipped with dual 2.4 GHz AMD Opteron 250 CPUs and 8 GByte of RAM.

Each model of a plant has been converted into a separate OpenRT object, containing the geometric primitives, references to shaders, and its own acceleration structure (see Section 4.1). Because not too many different models are used, all data can be loaded into main memory on every client PC (866 MByte in total), and are then referenced from their instantiations.

By adapting a few key parameters we can seamlessly trade-off between interactivity and accuracy. The most important parameters are the number of primary ray samples per pixel, the number of shadow rays or virtual light sources in the environment, and adaptive control of the maximum recursion for transparency. We generally maintain two sets of parameters: One low-quality set is used during interaction while the other high-quality set applies for still images

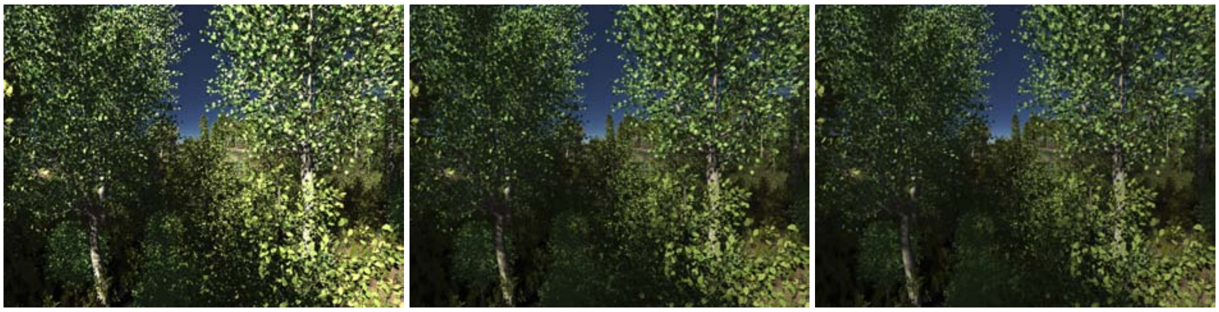

Figure 6: *Progressive refinement: An image with an increasing accumulation of intermediate results. a) No accumulation. b) After 4 accumulation steps. c) After 10 accumulation steps.*

that are incrementally computed when no user interaction occurs. Instead of computing a high-quality image in a single long computation, we iteratively accumulate samples computed over several frames. This is easily achieved through suitable initialization of random number generators across frames and gives frequent feedback to the user during convergence.

Figure 7 demonstrates the effect of interleaved sampling. In a) only a single primary sample per pixel has been taken each sampling only one virtual directional light. Note, that we still have to trace several rays per pixel due to the alpha textures. The term “sample” therefore describes a sequence of separately traced ray segments along one ray. Using 32 CPUs we achieve up to 6 fps at a resolution of 640×480 pixels.

An image with four primary samples per pixel is shown in Figure 7b). For each final hit point, four shadow rays are traced (and propagated through transparent surface parts). Due to interleaved sampling for each of the four hit points a different set of virtual light sources is used.

This results in a performance of roughly 1 fps. With a higher number of CPUs e.g. 48, the frame rate increases linearly (1.47 fps). The same is true when rendering these images on a PC with only a single CPU. Rendering the image with the same quality settings takes about 30 seconds. This level of performance is still adequate for many other applications that do not require full interactivity.

If approximated results are tolerable, interactivity can be further increased if adaptive transparency is activated. Figure 8 shows a comparison. In a) no restrictions on the maximum ray depth are imposed, while in b) leaf transparency is turned off for plants beyond a certain distance. For the transparent parts the average leaf color weighted by 0.75 is used instead of results from transparency rays. In both cases 48 CPUs were employed with an image size of 640×480 pixels. The frame rate increase from 2.1 fps in a) to 2.7 fps in b). Speedup in this case is only 29%, since the method can only be applied for far away objects. Note, that the results of this approximation tend to be too bright if no weight is applied, because later hit points are more likely to be inside of trees or plants and thus are more likely to be in shadow.

Finally, Figure 6 shows three different stages during progressive image accumulation. The left image shows the scene during camera motion (1 primary and 2 light samples). In the middle we see the scene after 4 images have been accumulated, and on the right after 10 accumulation steps. With a setup of 48 CPUs and an image size of 1270×960 pixels, it takes roughly 5 seconds (2 fps) until the converged result shown in image c) is computed.

6. Summary and Conclusions

In this paper we demonstrate initial steps towards realistic and interactive visualization of high-density plant ecosystems through fast ray tracing. By using highly efficient instantiations, optimized ray tracing engines are already capable of directly visualizing scenes containing more than 365,000 densely packed individual plants. The full model has been rendered without the need to simplify geometry or shading. The plug-an-play shading mechanism of ray tracing also allows for simple addition of advanced shading and lighting including environmental HDR lighting and pixel-accurate soft shadows.

However, in contrast to CAD models used in previous work on massive model visualization, the size and dense

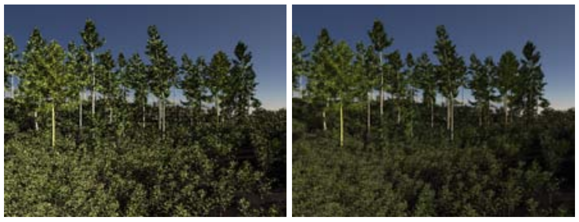

Figure 7: *Interleaved sampling. a) A single primary sample per pixel and only a single virtual light source is used. b) Four primary samples per pixel are taken. For each of the four hit points four shadow rays are fired, while a different set of light sources is applied for each hit point.*

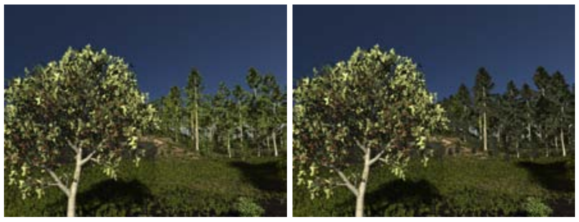

Figure 8: *Adaptive transparency. a) Computing all ray segments until a non-transparent part is hit. b) Transparency replaced by the weighted average leaf color for distant leaf geometry.*

structure of natural scenes required significant improvements to the rendering engine to achieve acceptable frame rates. Dense packaging, especially of small plant species, causes strong overlap between the bounding boxes of these plants. This significantly reduces the efficiency of the top-level spatial index and consequently reduces ray tracing performance. Even worse, countless small leafs textured with alpha-maps result in extremely high ray recursion levels of more than 30 segments per primary or secondary ray.

On a single PC this visualization system delivers high-quality images within 20-30 seconds but can be scaled linearly by simply adding more processing power. On a reasonably sized cluster of commodity PCs, interactive walk-throughs are easily possible. By applying discretization of the incident environmental light, random sampling of the resulting virtual light sources, and interleaved sampling, even photo-realistic images can be obtained at interactive rates at medium resolution. Straight-forward adjustment of the sampling parameters facilitates a convenient trade-off between interactivity and accuracy.

Using per-frame accumulation, high qualitative solutions can be generated progressively as soon as user interaction stops. This is especially helpful if a user prefers fast navigation, but does not need the full quality during movement. Still, depending on the settings and the available compute power, the image quality during interaction provides a very good impression of the final result. As a result, the current system is already usable for applications such as architectural planning or garden and landscape design.

7. Future Work

The challenge of faithful interactive visualization of complex landscapes opens up quite a number of new research directions. An obvious performance improvement would be a better integration between shading and ray traversal to avoid having to trace new rays from scratch at each transparent leaf, which could significantly reduce the current overhead.

Efficient anti-aliasing in such highly complex natural scenes is another promising direction. Standard adaptive algorithms cannot be applied here since the high geometric variance would trigger super sampling at essentially every pixel.

For a fully photo-realistic impression it would also be necessary to incorporate global illumination algorithms. This would account for indirect illumination from the environment as well as simulate the scattering of light within the trees caused by translucent leafs. This aspect is probably the most severe limitation of the current system.

On the modeling side it would be interesting to create scenes with significantly more detail. More diversity between plant species and instantiated plants would also improve realism. Furthermore, adding irregularities like broken branches and dirt would also help in increasing the level of realism.

Recently developed programmable hardware support for ray tracing [WSS05] promises to reduce the still significant hardware resources required for interactive use of this technology. However, the large working set of natural scenes requires further attention.

Acknowledgments

We would like to thank in particular Ingo Wald for providing access to the MPII PC cluster and for his help with shader programming. We also thank Dmitri Rubinstein, Kai-Uwe Renner, Johannes Günther, and *in*Trace GmbH. The project was partly funded by the Deutsche Bundesstiftung Umwelt (DBU) project "Lenné3D". The plant models were created using the Xfrog modeling software.

References

[ARBJ03] AGARWAL S., RAMAMOORTHI R., BELONGIE S., JENSEN H. W.: Structured Importance Sampling of Environment Maps. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* (2003), pp. 605–612.

[BCF*05] BEHRENDT S., COLDITZ C., FRANZKE O., KOPF J., DEUSSEN O.: Realistic Real-Time Rendering of Landscapes Using Billboard Clouds. In *Computer Graphics Forum* (2005). (Proceedings of Eurographics).

[Ben75] BENTLEY J. L.: Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM 18*, 9 (1975), 509–517.

[BWS03] BENTHIN C., WALD I., SLUSALLEK P.: A Scalable Approach to Interactive Global Illumination. In *Computer Graphics Forum* (2003), pp. 621–630. (Proceedings of Eurographics).

[CBL99] CHANG C.-F., BISHOP G., LASTRA A.: LDI Tree: A Hierarchical Representation for Image-Based Rendering. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1999), pp. 291–298.

[CCDH05] COLDITZ C., COCONU L., DEUSSEN O., HEGE C.: Real-time Rendering of Complex Photorealistic Landscapes Using Hybrid Level-of-Detail Approaches. *6th International Conference for Information Technologies in Landscape Architecture* (2005).

[DCSD02] DEUSSEN O., COLDITZ C., STAMMINGER M., DRETTAKIS G.: Interactive Visualization of Complex Plant Ecosystems. In *IEEE Visualization 2002* (2002), pp. 219–226.

[DDSD03] DÉCORET X., DURAND F., SILLION F. X., DORSEY J.: Billboard Clouds for Extreme Model Simplification. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* (2003), pp. 689–696.

[DHL*98] DEUSSEN O., HANRAHAN P., LINTERMANN B., MĚCH R., PHARR M., PRUSINKIEWICZ P.: Realistic Modeling and Rendering of Plant Ecosystems. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1998), pp. 275–286.

[DL05] DEUSSEN O., LINTERMANN B.: *Digital Design of Nature – Computer Generated Plants and Organics*. Springer, 2005. ISBN 3540405917.

[DN04] DECAUDIN P., NEYRET F.: Rendering Forest Scenes in Real-Time. In *Rendering Techniques 2004 (Eurographics Symposium on Rendering)* (2004), pp. 93–102.

[DVS03] DACHSBACHER C., VOGELGSANG C., STAMMINGER M.: Sequential Point Trees. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* (2003), pp. 657–662.

[Gla89] GLASSNER A.: *An Introduction to Ray Tracing*. Morgan Kaufmann, 1989. ISBN 0122861604.

[Jak00] JAKULIN A.: Interactive Vegetation Rendering with Slicing and Blending. In *Eurographics 2000 (Short Presentations)* (2000).

[KH01] KELLER A., HEIDRICH W.: Interleaved Sampling. In *Rendering Techniques 2001* (2001), pp. 269–276. (Proceedings of the 12th Eurographics Workshop on Rendering).

[KK86] KAY T. L., KAJIYA J. T.: Ray Tracing Complex Scenes. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1986), pp. 269–278.

[KK03] KOLLIG T., KELLER A.: Efficient Illumination by High Dynamic Range Images. In *Rendering Techniques 2003* (2003), pp. 45–50. (Proceedings of the 14th Eurographics Workshop on Rendering).

[Max96] MAX N. L.: Hierarchical Rendering of Trees from Precomputed Multi-Layer Z-Buffers. In *Rendering Techniques 1996* (1996), pp. 165–174. (Proceedings of the 7th Eurographics Workshop on Rendering).

[MB89] MACDONALD J. D., BOOTH K. S.: Heuristics for Ray Tracing using Space Subdivision. In *Proceedings of Graphics Interface '89* (1989), pp. 152–163.

[MDK99] MAX N., DEUSSEN O., KEATING B.: Hierarchical Image-Based Rendering using Texture Mapping Hardware. In *Rendering Techniques 1999* (1999), pp. 57–62. (Proceedings of the 10th Eurographics Workshop on Rendering).

[nat] NATFX: The bionatics home page. http://www.bionatics.com.

[Ney98] NEYRET F.: Modeling Animating and Rendering Complex Scenes using Volumetric Textures. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (1998), 55–70.

[PKGH97] PHARR M., KOLB C., GERSHBEIN R., HANRAHAN P.: Rendering Complex Scenes with Memory-Coherent Ray Tracing. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1997), pp. 101–108.

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The Algorithmic Beauty of Plants*. Springer, 1990. ISBN 0387946764.

[Pru] PRUSINKEWICZ P.: The virtual laboratory. http://algorithmicbotany.org/virtual_laboratory.

[RB85] REEVES W. T., BLAU R.: Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1985), pp. 313–322.

[RB93] ROSSIGNAC J., BORREL P.: Multi-Resolution 3D Approximations for Rendering Complex Scenes. In *Geometric Modeling in Computer Graphics*. Springer, 1993, pp. 455–465.

[RH94] ROHLF J., HELMAN J.: IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1994), pp. 381–394.

[SB87] SNYDER J. M., BARR A. H.: Ray Tracing Complex Models Containing Surface Tessellations. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1987), pp. 119–128.

[SD01] STAMMINGER M., DRETTAKIS G.: Interactive Sampling and Rendering for Complex and Procedural Geometry. In *Rendering Techniques 2001* (2001), pp. 151–162. (Proceedings of the 12th Eurographics Workshop on Rendering).

[SGHS98] SHADE J., GORTLER S., HE L., SZELISKI R.: Layered Depth Images. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1998), pp. 231–242.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* (2002), pp. 527–536.

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004. Available at http://www.mpi-sb.mpg.de/~wald/PhD/.

[WBDS03] WALD I., BENTHIN C., DIETRICH A., SLUSALLEK P.: Interactive Ray Tracing on Commodity PC Clusters – State of the Art and Practical Applications. In *Euro-Par 2003. Parallel Processing, 9th International Euro-Par Conference, 2003. Proceedings* (2003), pp. 499–508.

[WBS03] WALD I., BENTHIN C., SLUSALLEK P.: Distributed Interactive Ray Tracing of Dynamic Scenes. In *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics (PVG)* (2003), pp. 77–86.

[WDS04] WALD I., DIETRICH A., SLUSALLEK P.: An Interactive Out-of-Core Rendering Framework for Visualizing Massively Complex Models. In *Rendering Techniques 2004 (Eurographics Symposium on Rendering)* (2004), pp. 81–92.

[WFP*01] WAND M., FISCHER M., PETER I., AUF DER HEIDE F. M., STRASSER W.: The Randomized z-Buffer Algorithm: Interactive Rendering of Highly Complex Scenes. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (2001), pp. 361–370.

[WP95] WEBER J., PENN J.: Creation and Rendering of Realistic Trees. In *Computer Graphics (Proceedings of ACM SIGGRAPH)* (1995), pp. 119–128.

[WSS05] WOOP S., SCHMITTLER J., SLUSALLEK P.: RPU: A Programmable Ray Processing Unit for Realtime Ray Tracing. In *ACM Transactions of Graphics (Proceedings of ACM SIGGRAPH)* (2005).