EUROGRAPHICS 2005 / M. Alexa and J. Marks
(Guest Editors)

*Volume 24 (2005), Number 3*

# Realistic real-time rendering of landscapes using billboard clouds

S. Behrendt[1], C. Colditz[1], O. Franzke[2], J. Kopf[1], O. Deussen[1]

[1] University of Konstanz
[2] Dresden University of Technology

**Abstract**

*We present techniques for realistic real-time rendering of complex landscapes that consist of many highly detailed plant models. The plants are approximated by dynamically changing sets of billboards. Realistic illumination is approximated using spherical harmonics. Since even the rendering of simple billboard cloud plants is too time consuming, the landscape in the background is approximated with shell textures. The combination of these techniques allows us to render large scenes in real-time with varying illumination, which is interesting for computer games and interactive visualization in landscaping and architecture as well as modelling.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Image generation I.3.7 [Computer Graphics]: 3D Graphics and Realism

## 1. Introduction

It is still a challenge to realistically render outdoor scenes with interactive frame rates, since these scenes consist of a extensive number of highly complex geometric models. One square kilometer of forest contains millions of plants, hundreds of thousands of trees, and many small shrubs. Even the explicit storage of the plant positions would be tedious and must therefore be avoided. The vast amount of geometry can only be handled using efficient level-of-detail algorithms that approximate larger parts of the scenes with very simple elements.

In this paper this approach is demonstrated by representing models and parts of the scene through dynamically changing sets of billboards. After a survey of existing literature, we first present algorithms to approximate plant models using billboard clouds. The geometry is still needed for close-ups of the plants, since billboard clouds do not completely avoid parallax errors and cannot represent the fine details of a geometry-based model. To allow the approximation of larger parts of the scene, square tiles in the background consisting of many plant models, are represented by shell textures, e.g. sets of parallel semi-transparent billboards (Sec. 3). In Sec. 4 we introduce methods to realistically illuminate these billboards, which is important for a realistic rendering of the scene, and for a smooth transition from billboard approximation to the detailed geometry. Finally we will present and discuss results and give an outline of our future work. In the accompanying video we demonstrate how the combination of the methods enables us to render large scenes at interactive frame rates.

### 1.1. Related Work

Related work can be divided into three sub-areas: level-of-detail algorithms for tree models, billboard techniques and realistic rendering.

**Tree-specialized simplification:** There have been a number of approaches to speed-up the rendering of complex models. Many automatic methods for reducing the geometric complexity of surfaces by triangle decimation have been developed during the past years. For a survey see [HG97]. These methods can efficiently be applied to smooth objects. However, trees cannot easily be simplified with these techniques without changing the overall appearance, due to the complex topology.

As early as 1985, Reeves and Blau [RB85] represented trees by collections of disks in order to reduce the complexity of the foliage. Carefully selecting colors and shadowing effects resulted in very aesthetic 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, which were placed inside what they called masses (agglomerated leaves of the foliages), 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].

Shade et al. [SGwHS98] used layered depth images (LDI) to render complex natural objects from pre-computed pixel-based representations with depth at places in which full polygonal representations are too expensive, and impostors such as billboards, are too inaccurate. Chang et al. [CBL99] developed a hierarchical form of LDI, which allows them to generate different levels of detail. Both approaches are pixel-based, which results in a low image quality for close-ups. Max [Max96] models and renders trees hierarchically from pre-computed images with Z-buffers. A twig consists of some images of leaves, and a branch of several images of twigs. Looking 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 images of the first order branches. A more detailed representation is achieved by replacing the images of first order branches through several twig images, and so on. Although, Max et al. [MDK99] accelerate the process by using texture hardware, the rendering times are much above what is needed for interactive applications due to extensive texture transfer operations.

Meyer et al. [MNP01] use an image based representation consisting of a hierarchy of bidirectional textures (HBT), to interactively render a landscape covered by trees. Variations of view and lighting direction are supported by storing HBTs for many directions. Shadowing is performed by interpolating the visibility information from pre-computed visibility cube maps located at the corners of the object bounding boxes.

A professional software designed for interactive rendering of tree models is Speedtree [Spe]. Here, specialized models are created manually and are animated. The system includes a set of trees. However, each model has to be created from scratch, while we utilize an already existing large tree library consisting of highly detailed polygonal models [Gre], enabling us to use the same model basis for all of our representations.

Neyret [Ney98] converts complex natural objects into volumetric textures, which are then raytraced. Mip-mapping is used to reduce texture data, and to generate samples from trees in a distance. The method provides high quality images of very complex scenes at moderate rendering times of several minutes, but it is not applicable in interactive applications. Recently this technique was adapted to graphics hardware by the authors [DN04]. However, that approach requires storing two representations of a volume texture in GPU memory: a 3D texture and a stack of 2D textures. We use a similar technique to render larger parts of our scenes in the background, but actually we need only to store the 2D texture stack representation.

**Billoard-Representations:** In contrast to the above described 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. Therefore it can be used only in the background, since no parallax distortion can be handled. A better approximation is achieved by Jakulin using sets of parallel billboards to represent trees [Jak00]. However, artifacts occur due to the parallel orientation.

To represent tree models using billboards, the "billboard clouds" approach can be used [DDSD03]. It represents a geometry by a set of arbitrarily oriented billboards. However, for tree models the proposed algorithm does not work optimally, since in this case the used plane-space transform does not provide significant results due to the non-compact geometry of our tree models. In this work we also use billboard clouds, but our method to generate the billboard set is better suited for trees. We use a clustering algorithm on the basis of the model vertices that enables us to find better billboard approximations.

**Realistic Rendering:** This is still challenging, since the models, and therefore also the light interaction, are quite complex. Even a single tree consists of thousands of leaves, which scatter light. Franzke and Deussen provide raytracing algorithms to render trees in a botanically correct way [FD03]. A quite accurate method for offline rendering of landscapes covered with trees, was proposed by Qin et al. [QNTN03]. The plants are represented using a multi-layered data structure that makes it possible to represent direct and indirect lighting effects. However, the models are in part a two-dimensional approximation, which prevents their usage in a close distance to the virtual camera. In contrast to their approach, we use a three-dimensional representation based on sets of billboards and try to illuminate them. Here, a method provided by Sloan et al. [SKS02, SHHS03] is used that approximates complex reflection behaviour with spherical harmonics approximations of the reflection function.

Our approach avoids most of the restrictions of former systems. In the following we first describe how to represent a plant model using a set of billboards.

## 2. Billboard Approximation of Trees

As mentioned in the introduction, algorithms that can be applied to smooth geometry, cannot automatically be applied to plant models. The "billboard clouds" approach by Decoret et al. [DDSD03] allows to find nearly optimal sets of billboards for smooth objects by transforming all triangles in the scene into the dual plane-space, where they are represented as points. Subsequently, point clusters are determined and represented by billboards. Unfortunately, with tree foliage

(a) (b) (c) (d)

**Figure 1:** *Tree approximation: a) original model; b) billboard approximation using k-means algorithm; c) approximation using improved clustering and hierarchical model information; d) other approximations of plants using billboards.*

this procedure results in a nearly even distributed point set in which only weak clustering appears.

### 2.1. Clustering

We modified the idea of Decoret et al. and applied a clustering algorithm directly to the vertices of the triangles that constitute the tree model. Each of the clusters is then represented by one or more billboards in dependency of the form of the respective point cloud. The number of clusters can be specified by the user to control the rough number of billboards generated by the algorithm. We normally use $50-200$ billboards for trees and $20-100$ billboards for smaller plants, depending on the size and complexity of the plant. In Fig. 1(a)-(c) a tree geometry is shown together with two billboard approximations.

Using the standard k-means clustering [Fuk90], a suboptimal set of billboards is produced. Especially for the trunk and the tree skeleton, visual artifacts can be seen. This is due to the clustering that does not take into account any model-specific characteristics. Besides k-means, the isodata algorithm is often used for clustering [BH65]. In contrast to the k-means approach, here clusters are dynamically split or merged due to their size.

The best result can be achieved, if knowledge of the model is available. In our approach, we receive the triangles of the geometric description in a hierarchical form. All triangles that belong to one branch including its sub-branches, subsequently are stored in the file. We distinguish between branches and leaves by analyzing the texture information, more precisely the color and the ratio of transparent and opaque texels on the triangles. This allows us to use geometric descriptions of plants delivered by a number of modelling tools.

Using this information, the clusters can be oriented along the branches of the tree, which yields visually pleasing results. In Fig. 1(a)-(c) a flimsy tree is shown in order that all billboards can be seen. The user can specify a branching level, so that each sub-branch of that level forms a cluster. This gives often visually pleasing results as seen in Fig 1(c). In that case the second level was chosen.

In Fig. 1(d) two other models are shown. Also in these cases the algorithm finds good representations, even though the structural desription of the models is different.

The clustering can also be applied to sets of plants. In this case, a much coarser representation is produced; however, it can be used in the background of the scenes. This helps reducing the overall number of elements, which have to be rendered for a complex scene. Alternatively, we use shell textures to represent larger parts of the scene in the background (see Sec. 3).

### 2.2. Visual Representation of Clusters

To determine the form of a cluster that should be represented by billboards, an oriented bounding box is generated. Finding an appropriate box is not easy. In our experiments we first tried to work with a regression plane to which all points have a minimal squared distance. The plane then served as a basis for the bounding box. Unfortunately, the results are often visually not optimal. An attempt to use eigenvectors also

failed, since these vectors often do not represent the cluster in a visually optimal way. We discovered a better solution by calculating a bounding box with minimal volume. Analytic algorithms for determining such boxes exist; however, they work quite inefficient in the case of complex point clouds [BHP99]. Thus we developed a simple optimization procedure that jitters a given initial bounding box in order to find a better one. By rotating the box in one-degree steps, we achieved satisfactorily solutions efficiently.

After obtaining a bounding box for each point cloud, a special representation is chosen relative to its form. If the bounding box has two long sides and a short one, it is represented by a single billboard. If, however it has only one long side it is represented better by two crossed billboards. If all sides have roughly the same length, three crossed billboards are used. To decide the number of billboards the following heuristic is used: If $\sqrt{a \cdot b} > f \cdot c$ for the three sides $a$, $b$, $c$ then a billboard parallel to $a$ and $b$ is generated. We found that $f = 0.5$ produces visibly pleasing results. For each billboard, one texture is created for each side by projecting all geometry that is represented by the point cloud onto the billboards and rendering it.

### 2.3. Rendering

Rendering of the billboard textures can be performed in three variants: in the simplest form pre-lighting is used, and the billboards are created for a single light-position including all shadows. All billboard textures are stored in one texture atlas, which needs approximately 1 MByte of texture space. Normal mapping can be performed using special fragment and vertex shaders. The normals are stored per billboard texel in a separate normal map. During rendering the texture is obtained and the normals are used to shade the pixels of the billboards [DDSD03]. This method allows us to render the billboards for a variety of light positions and colors. Shadows must be computed additionally. In the third variant, which is described in Sec. 4, the reflection function is approximated using spherical harmonics basis functions. The parameters of these functions are also stored in additional textures. In this variant, nearly all lighting interactions can be stored including self shadowing of the objects. In the latter two cases the required texture memory for a typical tree model is $2-5$ MByte, which limits the amount of different models in a scene relative to the available memory on the graphics board.

The representation allows for the rendering of large landscapes covered by tens of thousands of trees (see Fig. 7). If the scene is rendered without the shell textures approach (see the next Section) we use a simple level-of-detail mechanism on the billboard clouds: we fade between a simple representation consisting of two crossed billboards and the complex version. In the background, only the simple representations are shown. We cannot simply fade the billboards out by modifying the alpha value, since then artifacts would occur if the single billboards are not rendered back-to-front. Hence, we use a different approach. We multiply the alpha channel of the billboard texture with a gauss curve in pre-processing. During runtime we use a custom fragment shader that implements a modified alpha test: a fragment is discarded, if its alpha value is below a threshold that is calculated by a simple piecewise linear function of the fragments distance to the viewer. After that the alpha value of the fragments that passed the test is set to one. This has the effect that the billboards fade out smoothly with increasing distance. Usually, we have to blend to the complex representation, if a typical tree is closer than 30 meters. Fig. 6 shows different steps in the transition from the simple representation to a set of 60 billboards.

## 3. Representing Areas

Although the billboard cloud approximation of individual plant models is simple, rendering of massive scenes remains difficult. Therefore we use a texture-based approach to display these large scenes. The main idea is to approximate the geometry of the vegetation with a 3D texture rather than directly displaying it (see [DN04]). As a result the performance is independent of the geometric complexity of the vegetation.

### 3.1. Texture Generation

The first step in finding appropriate texture-based representations for plant scenes is to convert the given models into 3D textures.

A straightforward method would be to overlay the geometry with the texture grid and to compute the color at each texel position by intersecting a gaussian filter with the geometry [LPFH01]. This method provides good results, but is not optimal since it does not account for occlusion. For arbitrary models, occlusion cannot be taken into account, because the direction from which the models will be viewed is not fixed.

However, with models that form a vegetation layer this is different. The viewer will usually be positioned above the vegetation, thus the filter should be gaussian only horizontally, but account for occlusion vertically. Such a filter can be approximated by standard rendering approaches [DN04]. Here the plant model is cut into slabs with a diameter of one vertical voxel layer. These slabs are then rendered one by one with an orthographical camera looking from above, into a set of 2D textures. The combined texture set then forms the 3D texture.

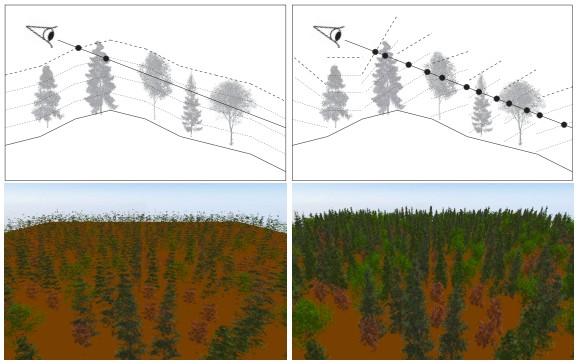
### 3.2. Rendering

Instead of using standard volume rendering techniques (see e.g. [EKE01]) that fail to render the massive texture sets that occur in our approach, we use slices that are offset surfaces of the terrain surface and represent the plant layer by a set of such slices. The main advantages of this approach are:

- the slice geometry can be generated on the fly from the ground geometry that is stored on the GPU. Hence, there is no need to transfer any geometric data for the slices per frame.
- the $y$-components of the sample coordinates have fixed values. Thus, the 3D texture can internally be represented as a set of 2D textures, which is more efficient on modern graphics boards. The increased performance is mostly due to the filtering that is bi-linear with 2D textures instead of tri-linear filtering with 3D textures. Furthermore, in our experiments we found that 2D textures are more cache efficient than 3D textures. Another advantage is that we are able to use anisotropic filtering with 2D textures, which is not possible with 3D textures on current graphics hardware.

The main difficulty in this approach is that with a small incidence angle very few samples could lie on a viewing ray. In the extreme case, when the viewing direction is parallel to the slice geometry there is no sample at all on the viewing ray. A simple way to solve this problem is to slant the triangles of the slices towards the viewer. Figure 2 shows the effect with strongly exaggerated angles.

Because we use 2D textures with fixed content, an error occurs using this method: the slanted triangles display the samples in the wrong position. However, this error is barely noticeable in our scenes, because we use a small slanting angle and the shell textures are used only in the background. Additionally the complexity of the plants helps to hide this error.



(a) (b)

**Figure 2:** *Viewing slices with small incidence angle. a) traditional display with low quality; b) slanting of slices.*

For the following calculation let $\vec{x}_1, \vec{x}_2, \vec{x}_3$ be the vertices of a triangle of a slice to be slanted and $\vec{c}$ the viewer position. Then $\vec{m} = (\vec{x}_1 + \vec{x}_2 + \vec{x}_3)/3$ is the center and $\vec{n} = (\vec{m} - \vec{x}_1) \times (\vec{m} - \vec{x}_2)$ the normal vector of the triangle. It is totally facing the viewer, if $\vec{v} = \vec{c} - \vec{m}$ and $\vec{n}$ are in parallel. This is the case, if

$$\vec{v} \cdot (\vec{m} - \vec{x}_i) = 0 \qquad (1)$$

Thus, to slant the triangle towards the viewer, we have to find new vertex coordinates $\vec{x}_i'$ so that Eq. (1) is satisfied. In this form there are infinite solutions, because there are no restrictions for the new positions. To receive a single solution $\vec{x}_i'$ is generated by a vertical shift: $\vec{x}_i' = \vec{x}_i + (0, d_i, 0)$. Now Eq. (1) is solved for $d_i$:

$$d_i = \frac{\vec{v} \cdot (\vec{m} - \vec{x}_i)}{v_y} \qquad (2)$$

However, with this solution, for low incidence angles $d_i$ increases to infinity. To prevent this, we limit the maximum slanting by modifying Eq. (2):

$$d_i = \frac{f \cdot \vec{v} \cdot (\vec{m} - \vec{x}_i)}{\|\vec{v}\|}$$

where the constant coefficient $f$ adjusts the maximum angle $\phi_{max}$ between the original and the slanted triangle, which is $\phi_{max} = \tan^{-1}\left(f \cdot \frac{\vec{v}}{\|\vec{v}\|} \cdot \frac{\vec{m}-\vec{x}_i}{\|\vec{m}-\vec{x}_i\|}\right)$. We used $f = 1$ for all images. With the modified equation, the displacement of the vertices is now proportional the cosine of the incidence angle $\angle(\vec{n}, \vec{v})$ and the distance from the vertex to the triangle center $\|\vec{m} - \vec{x}_i\|$.

The calculation is performed very efficiently in a vertex shader since the slice geometry is generated on the GPU. The only additional information needed per vertex is the triangle center, which is stored as a vertex attribute.

### 3.3. Level-of-Detail on Shell Textures

In distant regions the vertical distance between adjacent slices is very small in screen space, so that the triangles are nearly mapped onto each other. Thus, it is an obvious optimization to reduce the number of slices in distant regions by replacing adjacent slices with a single slice having a precalculated texture. In order to render any number of slices, we create a texture pyramid in a pre-processing step for the original texture stack [DN04], see also Fig. 3.

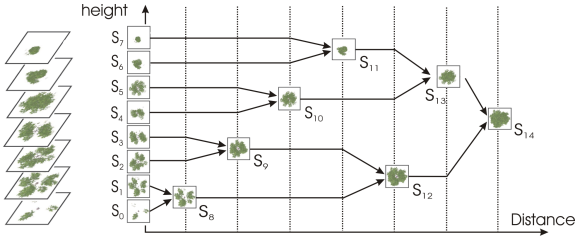



**Figure 3:** *Level-of-Detail on a set of slices. Distant objects (and regions) are rendered with only a single slice.*

The usage of the pre-calculated textures changes the order in which the textures are blended. If the standard blending formula $c'_{dest} = c_{source} \cdot \alpha_{source} + c_{dest} \cdot (1 - \alpha_{source})$ is used, blending is not associative and thus a slight change

in colors can be seen, as soon as the number of slices changes. In order to prevent this, we pre-multiply the color channels of the slices with the alpha channel, i.e. we store $(r \cdot \alpha, g \cdot \alpha, b \cdot \alpha, \alpha)$. In addition we use the modified blending equation $c'_{dest} = c_{source} + c_{dest} \cdot (1 - \alpha_{source})$. To simplify the following equations, we use the symbols $\bullet$ and $\circ$ for the standard and modified blending equation, respectively.

It can be easily verified that $c^{pre} \circ F \approx c \bullet F$, and that if rendering pre-blended slices $\left(c_0^{pre} \circ c_1^{pre}\right) \circ F \approx c_0 \bullet (c_1 \bullet F)$. The equal sign is not used, because the results with the original and modified blending equation differ in the alpha value of the resulting frame buffer content. Since the alpha value is not needed for display, using the modified blending equation produces in both cases – with or without pre-blended slices – the same result as the standard blending equation, if the single slices are blended one after another.

In order to achieve a smooth transition between two levels, we shift the slices vertically in dependence to the distance of the viewer. The two slices are moved closer together until they finally touch. Then they can be replaced by a single plane (see Figure 3). This calculation is also performed in the vertex shader. The vertical offset is piecewise linear and can thus be represented by an array of offsets. The Array can be accessed using the address register functionality.

In the region near the viewer, we blend the billboard cloud representation with the shell textures. We first render the shell textures and then render the billboard clouds near the camera. In a transition region, we then modify the alpha value of the shell textures and the billboard clouds, so that the shell textures fade out as the billboard clouds fade in. Theoretically this gives incorrect results, since the shell textures and the billboard clouds should be rendered back-to-front at the same time in the transition region. However, the artifacts due to our approach are negligible as can be seen in the accompanying video. In our scenes we use a linear fade from 20m to 100m distance to the camera.

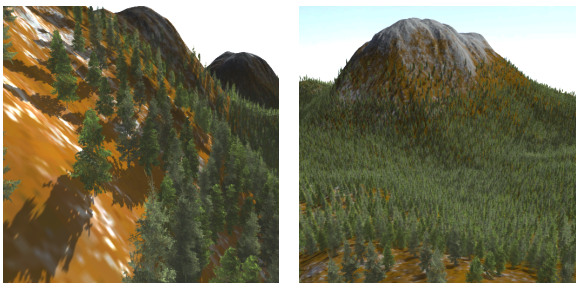


**Figure 4:** *Complex scene rendered using shell textures.*

This sampling scheme allows a level-of-detail rendering of vegetation using 3D textures without visible popping artifacts and low aliasing. However, the immense memory requirement of the 3D texture is still a great limitation. Even for small terrains, the local memory of current graphics hardware is not sufficient. We solve this problem by replicating a set of prototypical texture tiles over the terrain. For a smooth transition over tile boundaries, and to limit repetition artifacts, we use the Wang tiles approach [Sta97, NC99, CSHD03].

### 3.4. Rendering Using Wang-Tiling

Wang Tiles are square tiles whose edges are associated with a color out of a finite color set. In a valid tiling of the plane, the edge colors of adjacent tiles must match. This constraint leads to a simple stochastic tiling algorithm for aperiodic tilings (see [CSHD03] for details).

In order to use this approach in combination with rendering the vegetation layer, the terrain is tiled using a Wang tile set. For each tile a 3D texture is created, which fits across boundaries with matching edge colors. If this is done carefully and enough tiles exist, most artifacts from the tiling of the plane can be removed. However, the freedom in placing plants at arbitrary positions is limited, and the creation of good tile sets and tilings is non-trivial.

The access to the wang tiles is implemented using a fragment shader. The tile set is first stored in a texture atlas. The tiling itself is stored in a large two-dimensional index texture, which contains the offsets in the atlas for each tile. The fragment shader fetches the tile offsets using the $x,z$ world coordinates of a fragment. These coordinates also provide the relative offset in the tile. Using the combined information, the correct fragment can be fetched from the texture atlas.

## 4. Realistic Lighting

The basis for an accurate optical representation of the billboards were presented by Sloan et al. [SKS02]. In this work a method is proposed to render complex geometries in a dynamic low-frequency lighting environment by using a special representation for the reflection function. So-called spherical harmonic lighting projects the rendering equation onto an ortho-normal basis (using the Legendre polynomials) over a sphere. In a pre-processing step a set of coefficients is computed from the original object. During runtime the lighting environment is projected onto the basis, and is then used to illuminate the geometry by simply evaluating the scalar product of the coefficients. For diffuse light interaction a single vector of real numbers is associated with each vertex of the geometry. To minimize the amount of data, a clustering method ( [SHHS03]) can be used to combine similar coefficient vectors.

Neither the complexity of the BRDF (BSSRDF in case of translucent materials) nor the number of lights are limited. It is also possible to include indirect illumination in the spherical harmonic solution, while the runtime performance

**Figure 5:** *Upper row: polygonal tree (126,000 triangles), lower row: approximation by 60 Billboards with lighting.*

**Figure 6:** *Level-of-Detail of a model on the basis of illuminated billboards. The representation is blended between two and 60 billboards.*

**Figure 7:** *A complex scene under different lighting conditions.*

does not decrease at all. The only restriction follows from the number of bands used to calculate the coefficients. The more bands are used, the more floating-point numbers have to be stored and processed during runtime. This leads to a reduced performance, while higher frequencies in the lighting environment can be captured.

Fortunately, natural environments are illuminated by low-frequency lighting. Photons originated from the sun are scattered by the atmosphere or clouds, which leads to a diffuse lighting environment. Exploiting this fact, we can use just a few bands (2 or 3) to pre-process the plants. Interestingly this is also the case for shadows that are casted by the tree onto itself. The coefficients are then stored at each vertex and can be used to illuminate the original geometry.

During creation of the billboard representation, we store the Gouraud-interpolated vectors in an extra texture. The base color (unlit rendering) is also saved in a map. During the rendering of a billboard plant, the spherical harmonic lighting texture is used to re-light the base color of the approximated plant. Since the coefficients were calculated using the full plant, the billboard representation is illuminated with nearly the original complexity. This enables us to re-light the approximated simple geometry of the billboard plants in a very realistic way. The illumination can be calculated entirely on the GPU using a simple fragment shader. In Fig. 5 a tree model and its billboard approximation is rendered for different light positions.

This fact is very important in a level-of-detail context. Without a coherent lighting solution, visual artifacts appear when switching between different representations for the models. Using the same spherical harmonic coefficients for all levels, overcomes this problem and results in a coherent solution.

Since the solution on the basis of the spherical harmonics was calculated locally per plant, global lighting effects like shadows from one object onto another, are not yet included. To include shadow casting into our system, we use a shadow maps [Wil78].

## 5. Results and Discussion

Using our system, we receive interactive frame rates even for complex scenes that consist of many thousands of trees. In Table 1 some results are shown. They were generated using a 3 GHz Pentium 4 processor with nvidia FX6800 graphics board.

**Table 1:** *Frame rates of the system for varying scenes, resolution is* $800\times600$ *Pixels. A typical tree consists of* $2-40$ *billboards.*

| Scene | # trees | # polys | fps. |
|---|---|---|---|
| Fig. 7 | 6,600 | $12,000-50,000$ | $9-21$ |
| Fig. 8 | 21,300 | $50,000-300,000$ | $4-12$ |

Figures 8 and 9 show two additional screenshots of our scenes and indicate the visual quality of our approach. It turns out that the shell texture approach is suitable even for parts of the scenery that are relatively close to the camera. These textures can be used above $20m-100m$ from the virtual camera. This is demonstrated in the accompanying video. The overblending of the billboard-represented models can be performed smoothly, in both images all models are billboard-based.

Additional problems are caused by blending the billboard representation and the polygonal model. The full geometric complexity must be used in the direct vicinity of the virtual viewer up to a distance of about $5m-20m$ relative to the size of the plant. There can be hundreds of thousands of triangles in some square meters of a very dense outdoor environment. Consequently interactive frame rates are avoided. This can only be circumvented by switching to the billboard representation at very close distances, which sometimes causes parallax problems. In the future we will consider this detail problem before interactively rendering complex outdoor scenes with nearly no restriction to the scene size.

## 6. Acknowledgements

A major part of this work was funded by the Deutsche Bundesumweltstiftung (DBU) within the project Lenné3d (see http://www.lenne3d.de)

## References

[BH65] BALL G., HALL D.: *Isodata: A novel method of data analysis and pattern classification*. Tech. rep., Stanford Research Institute Tech. Rep. NTIS-AD-699616, Stanford, 1965.

[BHP99] BAREQUET G., HAR-PELED S.: Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. In *1999 Symposium on Discrete Algorithms Conf. Proc.* (1999), pp. 82–91.

[CBL99] CHANG C.-F., BISHOP G., LASTRA A.: LDI tree: a hierarchical representation for image-based rendering. In *SIGGRAPH 1999 Conf. Proc.* (1999), pp. 291–298.

[CSHD03] COHEN M. F., SHADE J., HILLER S., DEUSSEN O.: Wang tiles for image and texture generation. *ACM Trans. Graph. 22*, 3 (2003), 287–294.

[DCSD02] DEUSSEN O., COLDITZ C., STAMMINGER M., DRETTAKIS G.: Interactive visualization of complex plant ecosystems. In *IEEE Visualization 2002* (2002), IEEE, pp. 219–226.

[DDSD03] DÉCORET X., DURAND F., SILLION F. X., DORSEY J.: Billboard clouds for extreme model simplification. *ACM Trans. Graph. 22*, 3 (July 2003), 689–696.

[DN04] DECAUDIN P., NEYRET F.: Rendering forest

scenes in real-time. In *Rendering Techniques 2004* (June 2004), pp. 93–102.

[DVS03] DACHSBACHER C., VOGELGSANG C., STAMMINGER M.: Sequential point trees. *ACM Trans. Graph. 22*, 3 (July 2003), 657–662.

[EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proc. 2001 SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware* (2001), pp. 9–16.

[FD03] FRANZKE O., DEUSSEN O.: Accurate graphical representation of plant leaves. In *Plant Modelling and Applications* (2003).

[Fuk90] FUKUNAGA K.: *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[Gre] GREENWORKS: Home page of the xfrog modelling software. http://www. xfrog.com.

[HG97] HECKBERT P., GARLAND M.: Survey of polygonal surface simplification algorithms. SIGGRAPH 1997, course notes, 1997.

[Jak00] JAKULIN A.: Interactive vegetation rendering with slicing and blending. In *Eurographics 2000 Conf. Proc. (Short Presentations)* (2000).

[LPFH01] LENGYEL J., PRAUN E., FINKELSTEIN A., HOPPE H.: Real-time fur over arbitrary surfaces. In *2001 Symposium on Interactive 3D Graphics Proc.* (2001), pp. 227–232.

[Max96] MAX N.: Hierarchical rendering of trees from precomputed multi-layer Z-buffers. In *1996 Eurographics Workshop on Rendering* (1996), pp. 165–174.

[MDK99] MAX N., DEUSSEN O., KEATING B.: Hierarchical image-based rendering using texture mapping hardware. *1999 Eurographics Workshop on Rendering* (1999), 57–62.

[MNP01] MEYER A., NEYRET F., POULIN P.: Interactive rendering of trees with shading and shadows. In *2001 Eurographics Workshop on Rendering Techniques* (2001), pp. 183–196.

[NC99] NEYRET F., CANI M.-P.: Pattern-based texturing revisited. In *SIGGRAPH 1999 Conf. Proc.* (1999), pp. 235–242.

[Ney98] NEYRET F.: Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (1998), 55–70.

[QNTN03] QIN X., NAKAMAE E., TADAMURA K., NAGAI Y.: Fast photo-realistic rendering of trees in daylight. *Computer Graphics Forum 22*, 3 (2003), 243–252.

[RB85] REEVES W. T., BLAU R.: Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *SIGGRAPH 1985 Conf. Proc.* (1985), pp. 313–322.

[RH94] ROHLF J., HELMAN J.: IRIS Performer: A high performance multiprocessing toolkit for real-time 3D graphics. In *SIGGRAPH 1994 Conf. Proc.* (1994), pp. 381–394.

[SD01] STAMMINGER M., DRETTAKIS G.: Interactive sampling and rendering for complex and procedural geometry. In *Rendering Techniques 2001* (2001), pp. 151–162.

[SGwHS98] SHADE J., GORTLER S., WEI HE L., SZELISKI R.: Layered depth images. In *SIGGRAPH 1998 Conf. Proc.* (1998), pp. 231–242.

[SHHS03] SLOAN P.-P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph. 22*, 3 (2003), 382–391.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH 2002 Conf. Proc.* (2002), pp. 527–536.

[Spe] SPEEDTREE: Homepage of the speedtree software: www.idvinc.com/ html/speedtreert.htm.

[Sta97] STAM J.: *Aperiodic texture mapping*. Tech. rep., R046. European Research Consortium for Informatics and Mathematics (ERCIM), January 1997.

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *SIGGRAPH 1978 Conf. Proc.* (1978), pp. 270–274.

[WP95] WEBER J., PENN J.: Creation and rendering of realistic trees. In *SIGGRAPH 1995 Conf. Proc.* (1995), pp. 119–128.

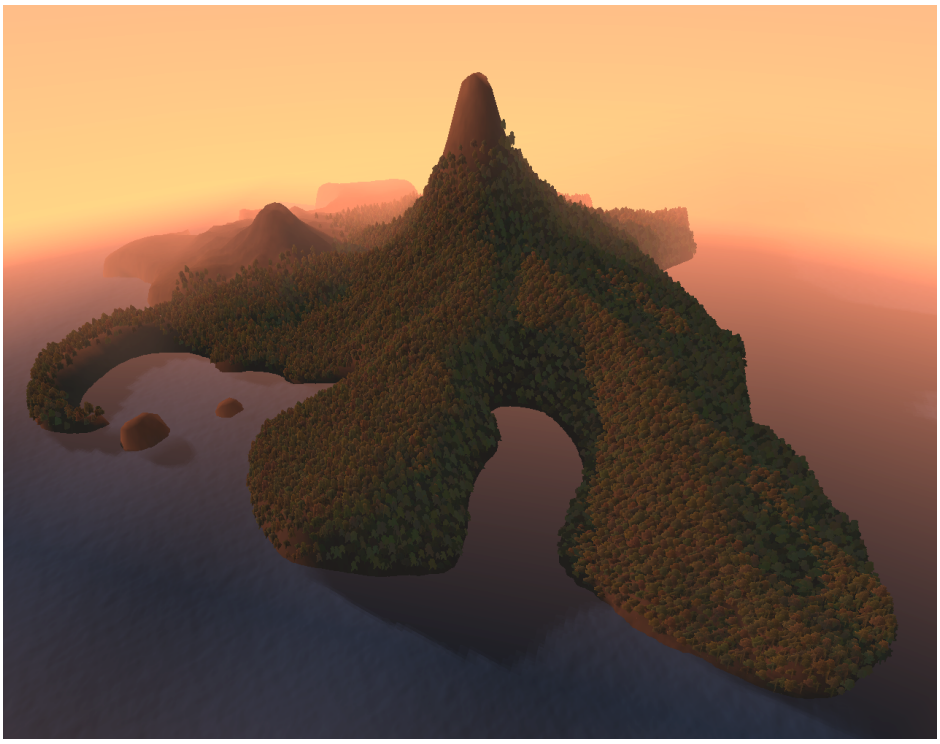


**Figure 8:** *Sample scene, rendered using billboard trees with approximate lighting.* $21,300$ *trees,* $4-21$ *fps.*

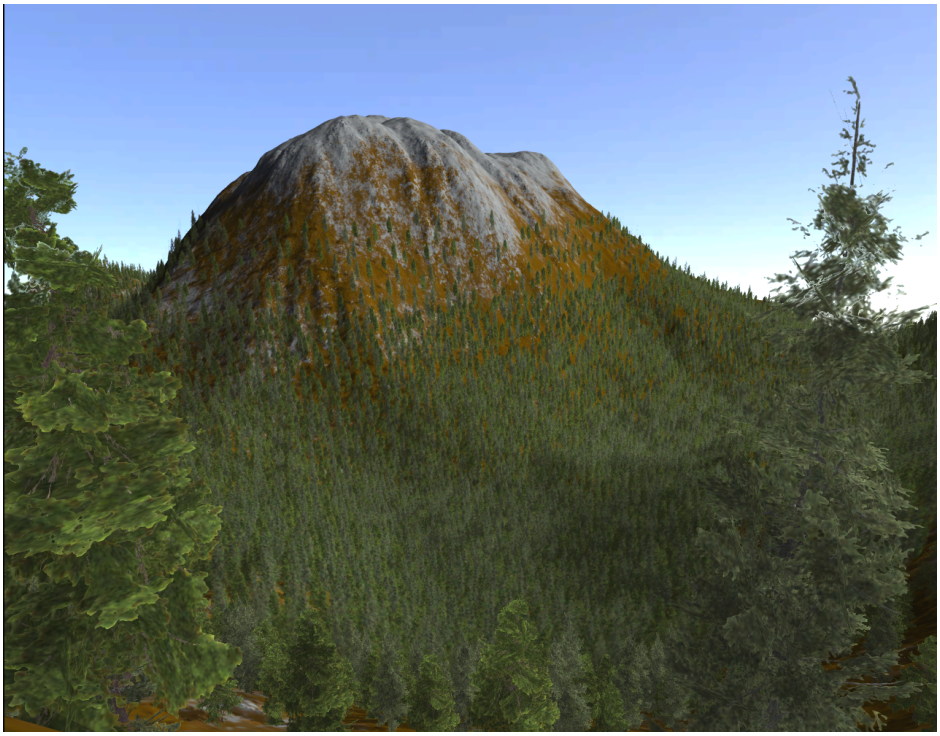


**Figure 9:** *Sample scene, rendered using shell textures in the background.*