Eurographics Workshop on Natural Phenomena (2007)
D. Ebert, S. Mérillou (Editors)

Multi-layered indirect texturing for tree rendering

Ismael García[1], Gustavo Patow[1], László Szirmay-Kalos[2] & Mateu Sbert[1]

[1]University of Girona, Spain
[2]Budapest University of Technology, Hungary

Abstract

This paper presents a technique to render in real time complex trees using billboard clouds as an impostor simplification for the original polygonal tree, combined with a new texture-based representation for the foliage. The technique provides several new contributions with respect to previous approaches. The new algorithm allows progressive level of detail both at the geometric and at the shader levels. It also preserves the parallax effects of the original polygonal model keeping leaf positions, orientations, and preserving the overlapping of the leaves as seen from any view point. In addition, the texture-based representation provides high-definition close views without introducing high memory requeriments. We adapted a realistic lighting model with soft shadows and a global illumination precomputation, allowing to render highly complex scenes with thousands of trees in real time.

Categories and Subject Descriptors (according to ACM CCS): http://www.acm.org/class/1998/ I.3.3 [Computer Graphics]: Image generation, I.3.7 [Computer Graphics]: 3D Graphics and Realism

1. Introduction

In computer graphics, one of the current most challenging problems is the generation and interactive rendering of vast natural scenes, involving hundreds of thousands of trees and other vegetal species. This is specially true in forest scenes, whose complexity is impossible to model in detail without efficient level-of-detail (LoD) algorithms and data structures. The original polygonal foliage models provide the natural parallax effect, but cannot be used in real-time visualizations of natural scenes with thousands of trees. On the other hand, most of the previous image-based foliage representations fail either to keep the parallax effect, provide low resolution detail, or fall back to wrong foliage alignment.

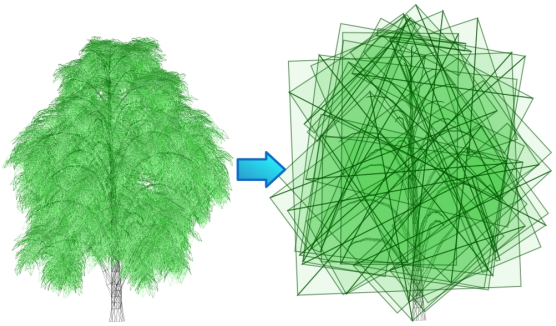
We propose a new texture foliage-based representation that provides high-definition from close views and progressive level-of-detail transitions with respect to the observer distance, keeping the appearance of the original complex geometry model and allowing realtime rendering of scenes with thousands of trees. In this sense, it outperforms existing techniques in the quality of close-views, with a much lower memory footprint. It also allows a progressive LoD technique that gracefully converges to the standard billboard cloud method for far-views, thus optimizing rendering efficiency. In addition, a lighting model including shadows, and ambient occlusion is incorporated to allow realistic, real-time visualization of large foliage sets.

The paper is organized as follows. In Section 2 we discuss previous work on interactive tree rendering. In Section 3 we present our foliage rendering approach. We describe the billboard cloud generation framework in Section 4 and we present in detail the visualization algorithm in Section 5. We give some implementation details in 6, demonstrate the results obtained using our techniques in Section 7 and discuss future work directions in Section 8.

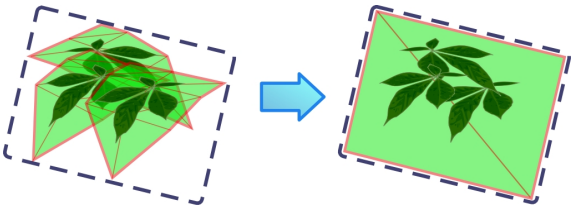
2. Previous work

The problem of rendering complex natural scenes has already received a lot of attention. There are two general approaches for interactive realistic rendering for trees: geometry-based (see [HG97] and [RCB*02]) and image-based techniques. Our work is included in the second group.

Image-based methods represent a trade-off between consistency and physical precision in favor of more photorealistic visuals. Billboard rendering is analogous to using cardboard cutouts: the billboard plane is always turned to-



(a) The polygonal leaves are replaced by a billboard cloud.



(b) Each group of leaves is replaced by a billboard.

Figure 1: *Replacing large groups of leaves by semi-transparent quadrilaterals with a texture map.*

wards the camera. Although this simple trick solves the parallax problem, the tree always looks the same no matter from where we look at it. Shade et al. [SGHS98] and Chang et al. [CBL99] used layered depth images (LDI) to render complex natural objects from pre-computed pixel-based representations with depth, with different levels of detail, but resulted in a low image quality for closeups. Max et al. [Max96, MDK99] modelled and rendered trees hierarchically from pre-computed images with Z-buffers, but the rendering times were not suited for interactive applications due to extensive texture transfer operations. Meyer and Neyret [MN98] converted complex natural objects into mip-mapped volumetric textures, which were then raytraced. On the other hand, in [RMMD04, LRMDM06] they estimated opacity in a volume, and then generated and displayed view-dependent textures attached to cells of that volume.

One of the most advanced methods actually implemented in commercial entertainment software [SPE05] is the basic free-form textured tree model, where the images are imposed on the approximated geometry. To represent tree models using billboards, the "billboard clouds" approach can be used [DDSD03], which represents a geometry by a set of arbitrarily oriented billboards. In our work we also use billboard clouds, but we generate the billboard set based on the work presented in [BCF*05] and [GSSK05], better suited for trees. We use a clustering algorithm on the basis of the leaf faces and vertices that enables us to find better billboard approximations. Then, leaves textures are used to add visual detail. In [GSSK05] it is also proposed an indirect texture approach, but it can only use one leaf image per cell, which leads to a regular-looking pattern that fails to preserve the original leaves positions and orientations and cannot preserve the original overlapping, resulting in sparser-looking trees, as seen in Figure 2. In [FUM05] they generated trees using super-sampling in an off-screen buffer, and then they applied downsampling with respect to a user-defined texture-size, trying to preserve the small details of the leaves. A drawback of this method is not being effective enough for close views.

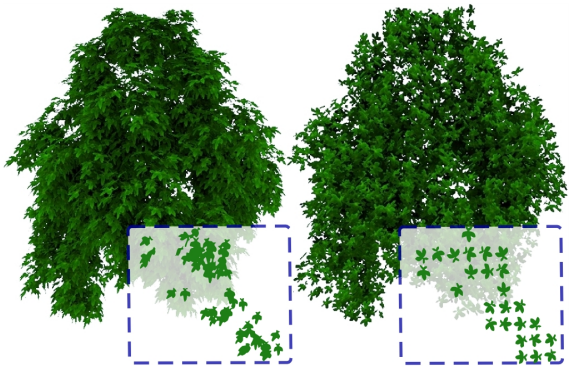


Figure 2: *Comparison and close up of the polygonal tree and billboards with simple indirect texturing [GSSK05].*

The techniques presented in this paper are related to the one presented in [Gla05], that placed small images at irregular intervals to help reducing the pattern artifacts of the free-form textured models, but did not handle the overlapping of the small images in a general case.

3. Overview

The presented method is based on replacing large groups of leaves by semi-transparent quadrilaterals defined as bill-

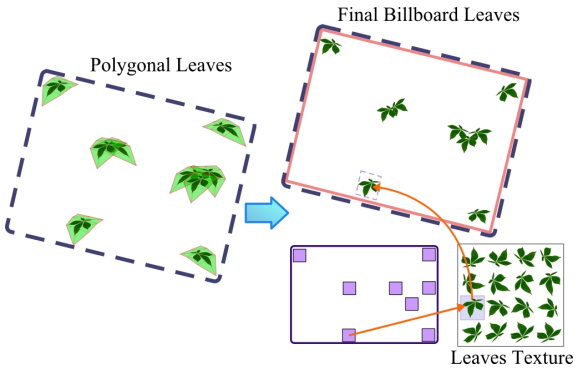


Figure 3: *Indirect texturing uses values in the texels of the indirect texture to fetch the corresponding leaf color image.*

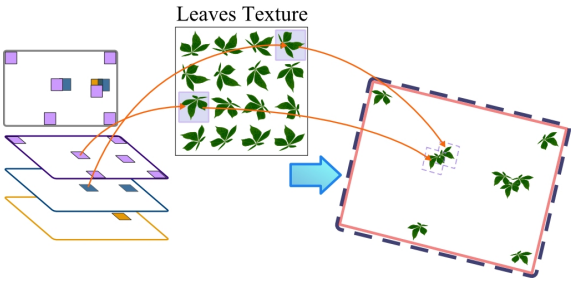


Figure 4: *Overlapped leaves must be encoded with a layered indirect texture to be properly rendered.*

boards (see Figure 1). Each billboard has a texture generated with an orthogonal projection of the polygonal faces of the leaves. In order to keep the leaves details, we use a procedural technique that replaces polygonal leaves by leaf patterns, called "indirect texturing" [OL98], which allows per-pixel computed values from one texture to be used as texture coordinates for an additional texture fetch. The leaves positions are used to place each leaf on the billboard surface, and an orientation index is used to fetch a leaf image from a color texture (called the Leaves Texture), as shown in Figure 3.

To properly handle the overlapping leaves within the same billboard, multiple texture maps would be needed, and the indirect texture should be encoded as a layered one, as shown in Figure 4. To improve storage and look-up efficiency, one indirect texture can be used to address lists of overlapping leaves, encoded in a second texture. Each entry in the lists contains a leaf position and an orientation index to address a leaf color image, as shown in Figure 5.

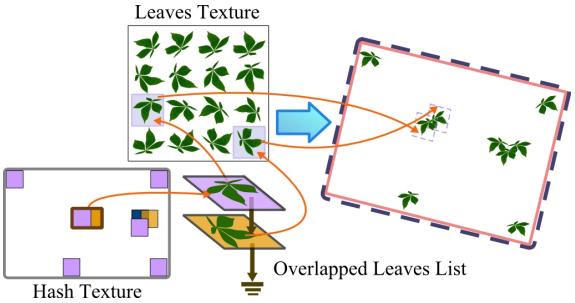


Figure 5: *Indirect texturing with overlapped leaf fragments lists reduce texture requeriments.*

The proposed method takes information from each group of leaves, generated by the billboard simplification algorithm and prepares a set of textures that are effective at different scales of simulation, from close views of individual branches and leaves, to bird's-eye flights, providing a viewing quality very close to the original complex polygonal model, as shown in Section 7.

4. Billboard cloud algorithm

As observed in Section 2, there are many automatic geometry simplification methods, but applying them to trees only provides acceptable results with the polygonal meshes that represent the trunk and the branches. Those methods do not work properly for the foliage. On the other hand, the billboard clouds approaches such as [BCF*05], can find nearly optimal sets of billboards for effective simplification, replacing sets of polygonal leaves by simple quadrilaterals.

4.1. Clustering

An algorithm like the one presented in [BCF*05] and [GSSK05] is used to group the leaves into clusters, where each cluster is represented by one billboard. One interesting property of the billboard cloud is that it completely bounds the shape of the original model, as shown in Figure 1.

The number of clusters can be specified by the user to control the number of billboards generated by the algorithm. The number of needed billboards depends on the size and complexity of the plant to be modelled. In general, the usage from 60 to 260 billboards for trees, or even fewer for smaller plants as bushes, is enough for visualization purposes.

4.2. Texture generation

Depending on the tree specie and age, a mature tree can have between 30,000 and 200,000 leaves. This intrinsic complexity leads to hundreds of leaves per billboard when a billboard cloud of about 260 quadrilaterals is used. As an example, one typical billboard plane, with 116 leaves, may have up to 10 overlapping leaves.

Our leaves are positioned with a continuous representation: each leaf has an associated vector that represents its position in local billboard coordinates (see Figure 8). In order to achieve an accurate overlapping, multiple layers must be used, so we subdivide the original billboard into a regular grid of cells, getting, for each grid cell, a set of cell-sized layers that may contain nothing (transparent) or a leaf fragment. Then, we link those cell-sized layers into a list of overlapping leaf fragments, as shown in Figure 6. We can consider that this grid of cells acts as a sort of spatial hash that accelerates fetches into this multilayered texture, so it is called the Hash Texture. The lists of leaf fragments are consecutively stored in a second, separate texture, which we call the Lists Texture (see Figure 8). Each list entry is a record that contains a reference to the third texture (the Leaves Texture), the leaf position vector, a scale factor, and any additional information that needs to be stored at the leaf level (see Section 5.2).

One of the key points of the proposed method to preserve accurate foliage placement, is the generation of the Hash and Lists Textures. After the clustering process, we send each leaf as a point sprite to an off-screen buffer, encoding the leaf

position and orientation and storing this information in the Lists Texture. To create the cell-sized layers, we use a depth-peeling technique [Eve99] that sorts the *leaf fragments* in front-to-back order, as shown in Figure 6.

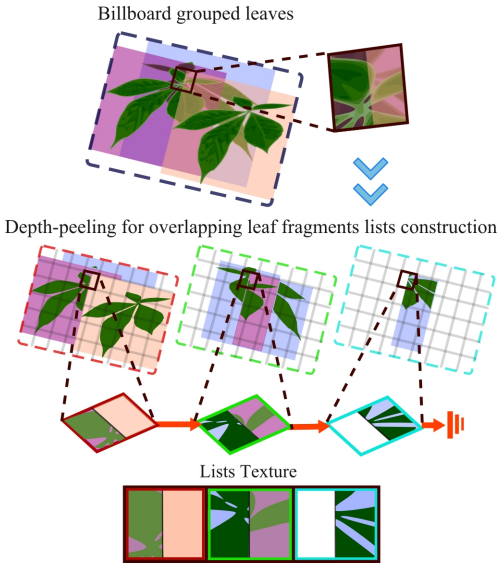


Figure 6: *Each leaf is processed as a point sprite. The point sprite can appear in more than one grid cell of the Hash Texture. The point sprites are processed with depth peeling to build an overlapping leaf fragments list for each grid cell.*

The size of the point sprite that holds the leaf information is defined with respect to its container billboard. This size is defined as the ratio between the 2D bounding box of the projection of the original leaf, and the size of the billboard (computed as $billboard_{pixels} = HashTex_{width} * HashTex_{height}$). In order to get the exact size of the point sprite in *pixels*, this size is multiplied by the total number of texels in the Hash Texture, following the expression:

$$leaf_{pixels} = \frac{area(leaf_{quad}) \cdot billboard_{pixels}}{area(billboard_{quad})}$$

leading to $leaf_{width} = leaf_{height} = \sqrt{leaf_{pixels}}$

The last preprocessing step builds the Hash Texture, where each texel addresses one entry in the Lists Texture, or transparent when the cell does not hold any leaf fragments. It is important to mention that the billboard polygons not only have *uv* texture coordinates associated with them, but also encode the Hash Texture resolution, for later use.

5. Visualization algorithm

When the graphics hardware draws a billboard, a single pass fragment shader is called with the corresponding *uv* texture coordinates. These coordinates are used to fetch the respective cell in the Hash Texture. As mentioned earlier, this cell can be either transparent, so the shader also will return a transparent fragment, or it can contain a reference to a list in the Lists Texture. Then, in an iterative process, each entry in the list is fetched until an opaque leaf fragment is found, or there are no more leaf fragments to evaluate. In the latter case, a transparent color is sent as output. This process is illustrated in Figure 7.

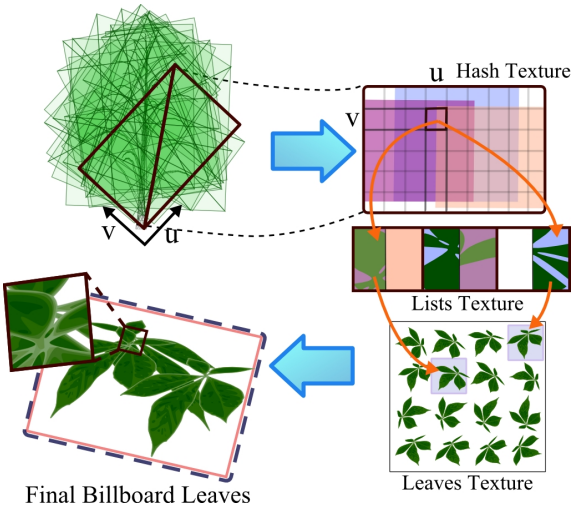


Figure 7: *Multi-layer indirect texturing evaluation in the single pass fragment shader.*

When evaluating each entry in the list, the reference to a leaf image into the Leaves Texture and the 2D vector giving the exact position of the leaf are retrieved. The position is subtracted from the *uv* coordinates the shader received to obtain a vector we call the Local Position vector. On the other hand, the leaf reference into the Leaves Texture is used as a vector pointing to the upper left corner of the corresponding image of the leaf. This vector is added to the Local Position vector to retrieve a texel from the Leaves Texture which either contains an opaque pixel with a color from the leaf, or transparent, indicating we indexed outside the leaf and we should continue with the next entry, as shown in Figure 8. If we add size information in the leaf entries, the Local Position vector is scaled accordingly.

Another important factor to take into account is that leaves overlap with a different order depending on the viewing direction. To incorporate this feature, the Hash Texture also stores the size of the list it is indexing, so we can access the list by both ends, just by looking at the direction of the billboard normal with respect to the viewer's direction. This also offers the advantage of allowing to index *two* Leaves Textures, one with the front sides of the leaves and the other with the back sides. These two textures can be unified in a single texture, and thus, a single indexing framework, to avoid further conditional branches.

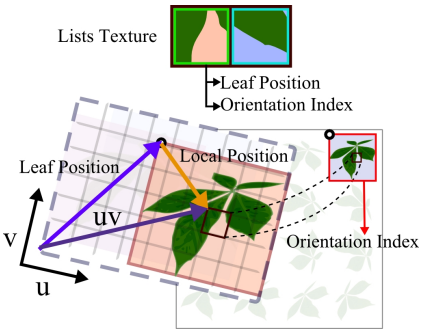


Figure 8: *Information stored at an entry in the list of leaves for a hash cell.*

5.1. Level-of-detail with indirect texturing layers

Although this method was designed for close or medium range views, it works well if mip-mapping is used for the Leaves Texture, as it would produce the right effect at large distances. However, the shader computations described above would be executed no matter the viewing distance, resulting in an unnecessary overhead that impacts on the fill-rate without representing a significative improvement in the visualization quality. To alleviate this problem, we introduce a progressive Level of Detail technique at the shader level.

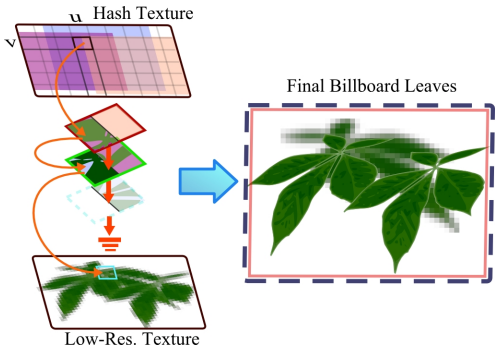


Figure 9: *As distance increases, the shader-LoD technique evaluates less list entries, and the missing ones are substituted by a low-resolution texture.*

Basically, this technique consists of reducing the number of evaluated list entries when the distance from the observer increases (see Figure 9). But evaluating less list entries means evaluating less leaf fragments, which will generate holes in the tree that get bigger as the viewing distance increases. This is solved by introducing a regular low-resolution texture that represents the missing leaves, generated in a pre-processing pass from the layered texture. So, as distance increases, less elements are evaluated and the missing ones are replaced by a single texture fetch into the low-resolution texture. Finally, when the viewing distance is high enough, no list entries are used any more and only the low-resolution texture is evaluated. Our experiments show that the fact that the leaves that are evaluated from a list entry are also present at the low-resolution texture does not produce any noticeable artifact. A comparison can be found in Figure 10. Finally, when the tree is at a very large distance, the technique would show only the low-resolution texture, without fetching any list. So, more evaluations can be avoided by a switch in the shader context to a shader that only evaluates the low-resolution texture without further computations.

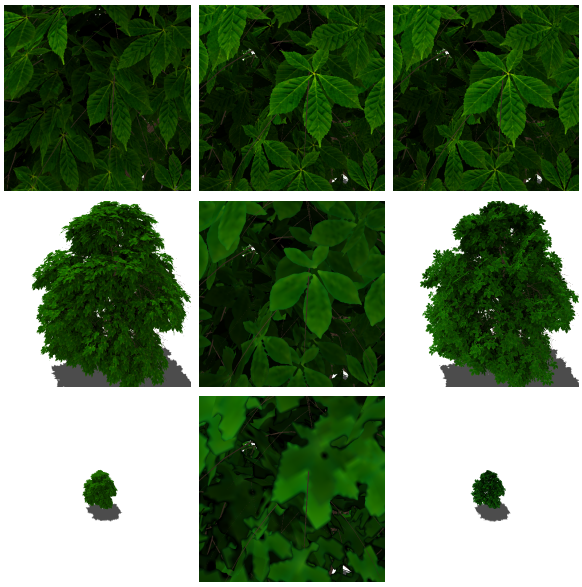


Figure 10: *Comparison between the polygonal tree on the left and the multi-layer technique on the right, preserving accurate alignment of the leaves. The middle column shows the progressive shader LoD. The small holes are due to slight changes in the leaves alignment.*

Fortunately, modern GPUs incorporate primitives (ddx, ddy) that help computing the number of needed evaluations. We have used the formula described by [LB06] (using the Hash Texture size stored in the billboard, as described in Section 4.2). Once we know the relationship between the Hash texel and its projection onto the screen, a number of needed entries to evaluate from the list is computed. This number is a factor times the length of the diagonal of the projected Hash texel. This expression is also used to choose the mip-map level of the Leaves Texture.

5.2. Photorealistic lighting

The shading model used for the trees can be split into two techniques. On the one hand, trunk and branches simplified models are rendered with bump mapping, ambient occlusion and shadow mapping. On the other, the foliage lighting is ob-

tained as the combination of the proposed multi-layer technique with ambient occlusion and shadow mapping.

Billboard clouds approximately preserve the original geometry, thus they can also replace the original tree when shadows or global illumination effects are computed. We also store a normal map together with the RGB map in the leaf texture, to allow for accurate real-time relighting. If needed, a correction for the billboard plane normal could also be stored in the entries of the Lists Texture, and used in conjunction with the local normal retrieved from the normal map to obtain a final normal to compute the shading.

To compute shadows, i.e., the visibility from point and directional lights (like the sun), we use Light Space Perspective Shadow Maps [WSP04]. To generate the shadow map, we adapted its usage in combination with the multi-layer method: for close-up views, where shadows need to be highly detailed, we use a full evaluation of the layers as described before. But, as soon as the observer gets farther enough, as described above, we switch to the low-resolution evaluation shader which provides good results at a fraction of the evaluation cost.

In order to complete the lighting model, we incorporate a pre-computed ambient occlusion [GFS07] term to take into account global illumination effects, computed for the original polygonal tree. We encode the ambient occlusion term as one new field in each entry in the Lists Texture. Because of the limitations of current graphics hardware for packing into single precision RGBA channels, we decided in our implementation to store the ambient term in a separate texture, indexed the same way as the Lists Texture.

To be compatible with our high-resolution representation of leaves into the Leaves Texture, we extrapolate the ambient term computed for the polygonal leaves to the four vertices of the 2D leaf bounding box. This is computed by setting a simple over-determined system of linear equations that relate the ambient term values at each projection of a polygonal vertex onto the billboard plane, with the corners that define the 2D bounding box of the projected leaf. See Figure 11. This system can be solved with any numerical method, or it can be approximated by taking only the four vertices of the original leaf that include the maximum value, the minimum value, and two extra values of the ambient term, and by solving only a system of 4 equations with 4 unknowns.

In the visualization, this ambient term is linearly interpolated in the fragment shader to provide the final ambient occlusion factor for the fragment on screen. One optimization would be, when the ambient term is stored in a separate texture, to use two consecutive rows to store the information for a single list, so ambient values would be interpolated by the GPU when queried. For this to provide good results, the Local Position vector mentioned in Section 5, should be used in combination with the texture coordinates used retrieved from the Hash Texture. For the low-resolution texture, we use an ambient occlusion term encoded in the alpha channel of the low-resolution leaves texture.

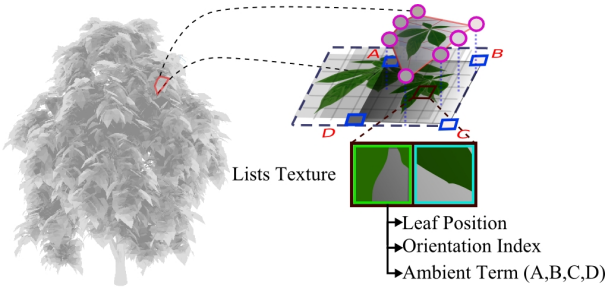


Figure 11: *Ambient term generation for the multi-layer leaves representation is encoded as a new field of the List Texture at the leaf level. The small squares represent the actual ambient occlusion values at the corresponding vertices.*

6. Implementation Details

The method presented so far requires one Hash Texture for each billboard, and in normal cases, there could be 256 and more billboards, largely exceeding current hardware capability of indexing textures at once. To handle this, several context switches would be needed, severely impinging on the overall performance. To solve this problem, all Hash Textures are condensed in a single large Hash Texture Atlas. In the same way, the different lists of the Lists Textures for each billboard are stored together in a single Lists Texture Atlas. In our experiments, the largest textures needed to store the Hash Textures and Lists Textures is about 1024^2 texels.

In addition to the shader level-of-detail technique described in Section 5.1, we used different billboard clouds for the foliage with a decreasing number of billboards, and switched among them as the distance to the observer increased. We stored three different billboard clouds that were stored in a single Vertex Buffer Object and indexed as necessary. The simplest billboard cloud is used in conjunction only with the low-resolution texture as described above, and the changes in billboard levels are ensured to be consistent with the corresponding shader LoD at the time of the switch. With respect to the tree trunk and branches, we used a progressive geometry-based Level-of-Detail technique, as described in [Hop96].

7. Results

The proposed technique has been implemented in OpenGL/Cg integrated into the Ogre3D engine, and run on a NV8800GTS graphics card. Figures for the usage of different tree species can be found in Table 1. The whole tree pre-processing stage takes between 2 to 5 minutes for all the examples presented here. All scenes were rendered at

Tree Species	Leaves	Polys/ leave	Billboards[3]	Memory[3]
Chestnut[1]	11291	10	278	3486762
Oak[2]	23660	2	194	3119734
Cork[1]	20144	4	121	1462870
Horse Chestnut[1]	9366	18	202	3376653
Shrub[2]	2388	2	198	1349091
Maple[1]	3662	2	175	1381600
Alder[1]	1049	2	228	1332837

Table 1: *Figures for different species showing the number of leaves, polygons per leaf, billboards and memory usage (in bytes) with our method. Labels mean: [1]Generated by Xfrog [Xfr06], [2]Generated by Forester, [3]Using texture compression DTX1-5, which provides a 1:6 compression ratio.*

Trees	Polygonal	Billboard
1	180	80
50	47	46
100	15	39
1000	5	28
10000	n/a	20

Table 2: *FPS for varying numbers of trees, for the original tree and the new method. Distances range from very close views to very far trees (covering only about 32^2 pixels).*

1280x1024 pixels, resulting in the frame rates of the system for varying number of trees presented in Table 2.

In Figure 12 we can see the results when compared to the original polygonal tree. As we can see in the figure, with only 278 billboard planes with a Hash Texture of 1024×1024 cells and up to 10 layers, the final results look very similar to the original. Figure 13 show complex scenes with tens of thousands of trees under very different lighting conditions and viewing distances.

8. Conclusions and future work

We have presented a method that allows interactive visualization of large forests, with tens of thousands of trees at interactive frame rates, even with realistic lighting effects like shadows and an ambient occlusion term. The new method consists of an indirect texturing technique, in combination with a texture used as a hash for fast indexing and retrieval, and a layered representation of the overlapping leaf fragments. Previous methods require extremely large texture memory space to preserve small details in the leaves, while the presented method preserves them with a low memory footprint and a few extra texture accesses. As an example, a chestnut would require about 246MB of texture space to achieve the same visual quality as our method, which only needs 3.7MB. For comparison, the original polygonal model for the leaves needs 23Mb.

Level-of-detail techniques have specifically been developed, not only at the geometric level (reducing the number of billboards and simplifying the geometry of the trunk and branches), but also at the shader level, drastically reducing the computational costs associated with rendering trees far from the observer. It is important to note that detail in the leaves is preserved even at very short distances, something that was not done before with billboard clouds. These results hold even for trees without dense foliage, making our technique highly suitable for instancing.

It must be mentioned that this algorithm is resolution dependant: depending on the screen-size of the trees and the screen resolution, different frame rates would be obtained. Nevertheless, as mentioned in Section 7, we are able to present about ten thousand trees in a complex scene in resolutions of 1280×1024 with an acceptable frame rate.

One line of promising immediate research is to change the progressive level-of-detail technique described in section 5.1 to a continuous LoD technique. This technique would perform an interpolation between shader levels, resulting in a smoother switch between LoD levels. Another interesting line is to extend the presented technique to include small branches, which would reduce the complexity of the geometry of the trunk, which then would be more easily handled with more traditional techniques. Finally, the incorporation of branch movements seems an interesting challenge to be modelled with the presented algorithm, as the Hash Texture would be slow to regenerate.

9. Acknowledgments

This work has been supported by GameTools FP6 (IST-2-004363) project, TIN2004-07451-C03-01 and TIN2004-07672-C03-01 projects from the Spanish Government, and by the Spanish-Hungarian Fund (E-26/04).

References

[BCF*05] BEHRENDT S., COLDITZ C., FRANZKE O., KOPF J., DEUSSEN O.: Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum 24*, 3 (2005), 507–516.

[CBL99] CHANG C.-F., BISHOP G., LASTRA A.: LDI tree: A hierarchical representation for image-based rendering. *ACM Computer Graphics*, Annual Conference Series (1999), 291–298.

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

[Eve99] EVERITT C.: Interactive order-independent transparency, 1999. White paper, NVIDIA Corporation.

[FUM05] FUHRMANN A. L., UMLAUF E., MANTLER S.: Extreme model simplification for forest rendering. In *Eurographics Workshop on Natural Phenomena* (2005).

[GFS07] GONZÁLEZ F., FEIXAS M., SBERT M.: An information-theoretic ambient occlusion. In *International Symposium on Computational Aesthetics* (2007).

Figure 12: *Comparison between the polygonal tree and the multi-layer technique. From left to right: Original polygonal tree fully lit, the wire frame billboards, billboards with diffuse lighting, direct lighting, shadows and, finally, the full illumination model including the ambient occlusion term.*

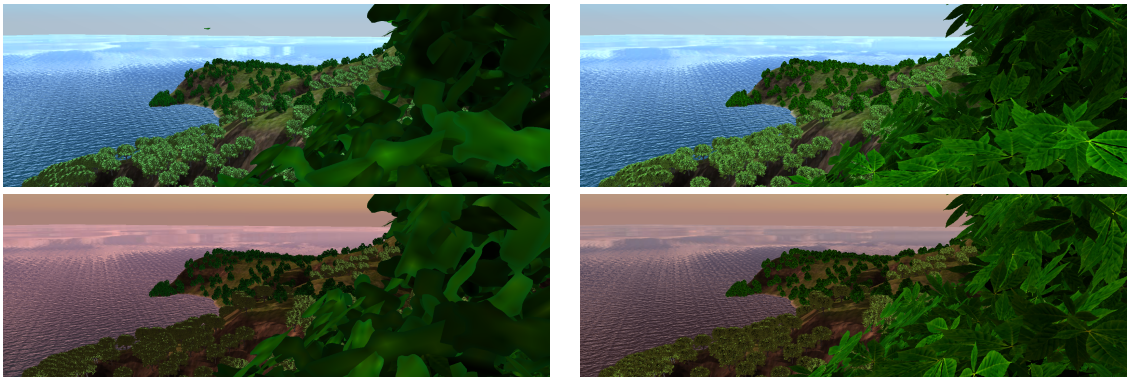


Figure 13: *A complex sample scene under different lighting conditions. Left column shows the scene with the classic method [BCF*05, FUM05] using the same memory footprint as the images on the right, using the new method.*

[Gla05] GLANVILLE R. S.: Texture bombing. In *GPU Gems 1*, Fernando R., (Ed.). Addison-Wesley, Oct. 2005, pp. 323–338.

[GSSK05] GARCÍA I., SBERT M., SZIRMAY-KALOS L.: Leaf cluster impostors for tree rendering with parallax. In *In Proceedings of EG Short Presentations 2005* (2005), pp. 69–72.

[HG97] HECKBERT P. S., GARLAND M.: Survey of polygonal surface simplification algorithms. In *ACM SIGGRAPH 1997 Course Notes* (1997).

[Hop96] HOPPE H.: Progressive meshes. *ACM Computer Graphics 30*, Annual Conference Series (1996), 99–108.

[LB06] LOVISCACH J., BREMEN H.: *Game Programming Gems 6*. Charles River Media, Inc., 2006, ch. Rendering Road Signs Sharply.

[LRMDM06] LINZ C., RECHE-MARTINEZ A., DRETTAKIS G., MAGNOR M.: Effective multi-resolution rendering and texture compression for captured volumetric trees. In *Game-On 2006* (2006), pp. 16–21.

[Max96] MAX N.: Hierarchical rendering of trees from precomputed multi-layer z-buffers. In *Eurographics workshop on Rendering techniques '96* (1996), pp. 165–174.

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

[MN98] MEYER A., NEYRET F.: Interactive volumetric textures. In *Rendering Techniques (Eurographics Workshop on Rendering)* (1998), Drettakis G., Max N., (Eds.), Springer Wein, pp. 157–168.

[OL98] OLANO M., LASTRA A.: A shading language on graphics hardware: the pixelflow shading system. *ACM Computer Graphics 32*, Annual Conference Series (1998), 159–168.

[RCB*02] REMOLAR I., CHOVER M., BELMONTE O., RIBELLES J., REBOLLO C.: Geometric simplification of foliage. In *Eurographics'02 Short Presentations* (2002), pp. 397–404.

[RMMD04] RECHE-MARTINEZ A., MARTIN I., DRETTAKIS G.: Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph. 23*, 3 (2004), 720–727.

[SGHS98] SHADE J. W., GORTLER S. J., HE L.-W., SZELISKI R.: Layered depth images. *ACM Computer Graphics 32*, Annual Conference Series (1998), 231–242.

[SPE05] Speedtree, interactive data visualization inc., 2005. http://www.idvinc.com/speedtree.

[WSP04] WIMMER M., SCHERZER D., PURGATHOFER W.: Light space perspective shadow maps. In *Rendering Techniques 2004 (Eurographics Symposium on Rendering 2004)* (2004), pp. 143–151.

[Xfr06] Greenworks: Organic software, 2006. http://www.greenworks.de/.

Figure 12: *Comparison between the polygonal tree and the multi-layer technique. From left to right: Original polygonal tree fully lit, the wire frame billboards, billboards with diffuse lighting, direct lighting, shadows and, finally, the full illumination model including the ambient occlusion term.*

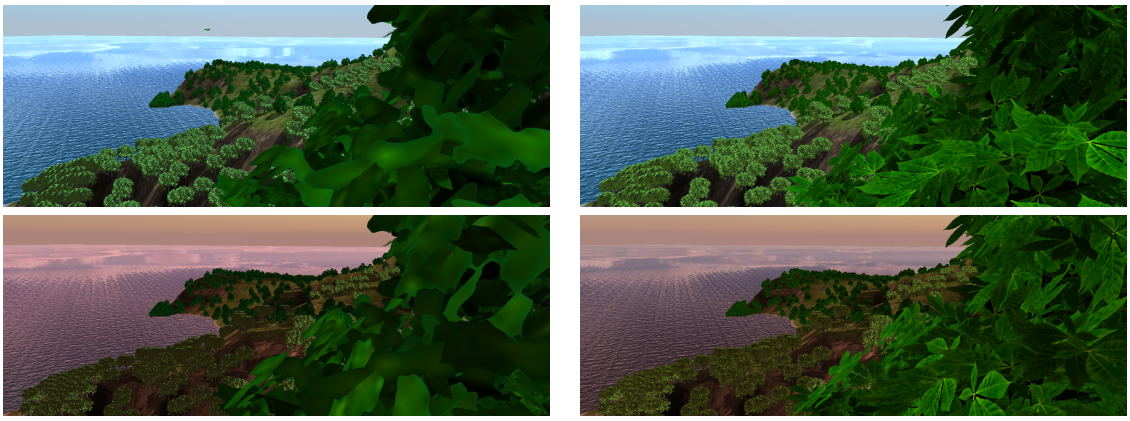


Figure 13: *A complex sample scene under different lighting conditions. Left column shows the scene with the classic method [BCF*05, FUM05] using the same memory footprint as the images on the right, using the new method.*