An efficient continuous level of detail model for foliage

C. Rebollo
Universitat Jaume I,
Castellón, Spain
rebollo@uji.es

I. Remolar
Universitat Jaume I,
Castellón, Spain
remolar@uji.es

M. Chover
Universitat Jaume I,
Castellón, Spain
chover@uji.es

O. Ripollés
Universitat Jaume I,
Castellón, Spain
oripolle@uji.es

ABSTRACT

Outdoor scenes require vegetation to make them look realistic. Current hardware cannot afford real-time rendering of these scenes because of the large number of polygons. Multiresolution modelling has been successfully presented as a solution to the problem of efficient manipulation of highly detailed polygonal surfaces. This article describes a new continuous multiresolution hardware-oriented model that can represent tree foliage with different levels of detail. The multiresolution model presented in this paper, *Level of Detail Foliage*, takes advantage of the programmable rendering pipelines nowadays available in most video cards. The geometry of the foliage is divided into a number of clusters, in some of which the detail can change while the rest of the clusters remain unaltered. This division of the foliage remarkably diminishes the number of vertices sent to the graphics system because only the information of the changed clusters are updated. The independent clusters condition a data structure that makes the time required for visualisation of the foliage more efficient. Here we present the data structure and the retrieval algorithms, which favour the extraction of an appropriate level of detail for rendering.

Keywords: Tree visualisation, interactive visualisation, multiresolution modelling, level of detail, hardware-oriented data design, clustering.

1 INTRODUCTION

Many of the interactive applications currently available such as flight simulators, virtual reality environments or computer games take place in outdoor scenes. In these environments, the vegetation is one of the essential components. The lack of trees and plants can detract from their realism. Nowadays, research on representation of plants has made possible to obtain very realistic models formed by a vast amount of polygons (Figure 1). This is a drawback for obtaining real-time visualisation.

Several methods have appeared up to now to solve this problem. They can be classified in two main groups: image and geometry based rendering. The geometry-based approach do not lose realism even when the camera is extremely near the object. Besides, this approach makes it possible to take advantage of the current graphics hardware, obtaining shadows or different ilumination effects. Other advantage is that geometry can be stored either in the main memory or directly in the graphics hardware, producing great acceleration in rendering.

Some techniques have been used in order to achieve interactive visualisation of very detailed objects. Multiresolution modeling has proved to be a good method as it adapts the geometric detail of these objects to the capacity of present-day graphics systems. These models typically work with general meshes. Nevertheless, Remolar et al. [16] presented in their work a multiresolution method especially designed to deal with the isolated polygons that form the foliage. (Figure 2).

Figure 1: Tree generated with the Xfrog commercial tool [22].

In this article, it is presented an efficient implementation of this multiresolution model, called *Level of Detail Foliage* LoDF. Its data structure has been designed as hardware-oriented in order to obtain a fast visualisation of the data. Besides, LoDF allows us to change the uniform level of detail in a continuous way. The basic idea of LoDF is to group the leaves in a number of independent clusters. This division makes it possible to exploit the capabilities of the latest hardware. All the geometric data is initially stored in the graphics card. In every change of level of detail, only the information about

WSCG 2006 conference proceedings, ISBN 80-903100-7-9
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.

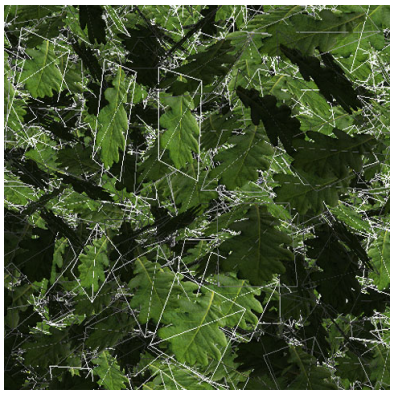

Figure 2: Detail of the foliage in a tree.

the clusters affected is updated and sent to the graphics system. The rest of clusters remain unchanged. This considerably reduces the traffic of data through the PCI bus and accelerates the visualisation process.

The article presents the following structure. Section 2 offers a brief review of the most important works in the visualisation models that have appeared up to now. In section 3 LoDF is analysed in depth, detailing the data structures it uses to store the information and the algorithms required to access these structures in order to retrieve that information in the most efficient manner. The visualisation process is proposed in section 4. Section 5 compares the results obtained with LoDF against the obtained using the model *View-Dependent Foliage* VDF presented by Remolar et al. [16]. Lastly, section 6 presents the conclusions and our future lines of work are outlined.

2 RELATED WORK

Research into real time visualisation of detailed plants is aimed at adapting the number of polygons used to represent the plants to the requirements of graphics hardware. As it has been previously said, these works can be grouped into two broad directions, depending on the method used to represent them. These can be works that use images or works that use geometry to represent the plants.

Image-based rendering is one of the most popular methods to represent trees because of its simplicity. Impostors [8] are the most common example of this approach. All the geometry that forms the tree model is previously rendered so as to obtain an image of it. This image is textured on a polygon using transparencies for correct immersion in the scene by replacing the geometric object. This way of representing vegetation has many problems, like the lack of parallax or the invariability of the rendered image when the camera changes its position. Some authors, such as Shade et al. [17], Marshall et al. [9] and House et al. [5], divide the scene into zones depending on the distance from the object to the viewer. Objects far away from the camera are represented by an image and objects near the viewer are depicted by geometry. Max [10] and Shade et al. [18] add depth information to the precalculated images. Other authors obtain 2D images from volumetric textures and combine them depending on the position of the camera. Meyer et al. [11] and Decaudin et al. [1] focused their work on forest visualisation, while Jakulin [6] and Reche et al. [13] were more interested in the representation of a single tree. Garcia et al. [3] solve the parallax problem using textures that group sets of leaves. One of most important commercial applications that uses this method to visualise trees is SpeedTree [19], which uses *billboards*, that is, impostors that are always oriented towards the viewer.

Regarding geometry-based rendering, the number of polygons that form the tree objects makes it necessary to use certain techniques to obtain interactive visualisation. Most of the works published to date change the display primitive using points or lines. Reeves and Blau [14] use particle systems to render trees in a forest, representing them with little circles and segments of straight lines. Weber and Penn [21] introduce level of detail techniques. In their work, leaves are represented with points while branches and limbs are represented with lines. The number of primitives that are displayed can be adapted depending on the size of the tree in the final image. Stamminger et al. [20] use points to represent trees in their work, changing the point density in real time depending on the relevance of the tree in the scene. Deussen et al. [2] and Gilet et al. [4] combine point and line representation with geometric representation. Their basic idea is to keep constant the number of vertices used to represent the forest. Trees near the camera are represented by geometry and trees situated far away from it are shown by points and lines.

In recent years, several works based on multiresolution models have appeared. Meyer et al. [12] and Lluch et al. [7] use a representation based on multiresolution models of images while Remolar et al. [16] deals with the geometry of the trunk and the foliage separately. They present a multirresolution model that adapts the number of leaves that form the tree in real time.

3 MULTIRESOLUTION MODEL

In general, a multiresolution model representation is constructed from two main elements: the original geometry of the object F_0, and the different approximations given by a simplification method, $F_1, F_2, ..., F_{n-1}$. The multiresolution scheme presented in this article is based on the Foliage Simplification Algorithm (FSA) [15]. The simplification operation of this method is *leaf collapse*: two leaves are collapsed into a new one.

LoDF uses two operations in order to increase or decrease detail in the approximation: *leaf split* and *leaf collapse* (Figure 3). The leaf-collapse operation diminishes the detail of the current approximation because it

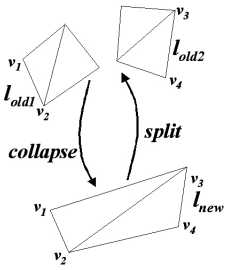

Figure 3: Example of leaf collapse and split operations.

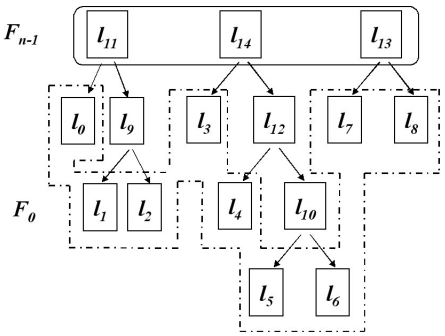

Figure 4: Example of data organisation in a F^r representation.

eliminates two leaves and adds a new one. In the example, leaves l_{old1} and l_{old2} will be eliminated and l_{new} added to the new approximation. However, the leaf-split operation adds detail to the representation being visualised. It eliminates one leaf l_{new} from the current approximation and adds the two leaves that were previously simplified, l_{old1} and l_{old2}.

$$Collapse(l_{old1}, l_{old2}) = l_{new} \quad (1)$$

$$Split(l_{new}) = (l_{old1}, l_{old2}) \quad (2)$$

The sequence of leaf-collapse operations obtained from FSA is processed to build the new multiresolution representation F^r. Each simplification operation creates one new leaf, numbered sequentially. In the case of a foliage with 9 leaves, labelled from l_0 to l_8, the first collapse operation creates the leaf l_9, the next simplification operation the leaf l_{10}, etc. One example of this data organisation is shown in Figure 4. The data are organised as a forest of binary trees, where the root-nodes are the leaves that form F_{n-1}, the coarsest approximation, and the leaf-nodes are the leaves of the original tree model, F_0. In this example, F_0 is formed by 9 leaves, and F_{n-1} by 3 leaves.

In the least detailed level F_{n-1}, several leaves from F_0 are represented by an only leaf. All of the leaves that form a binary tree in the data organisation are grouped in clusters in LoDF. In that way, foliage can be divided into independent groups. These clusters determine the data structure used in our model. Following the data structure shown in Figure 4, the groups that form this example are shown in Figure 5.

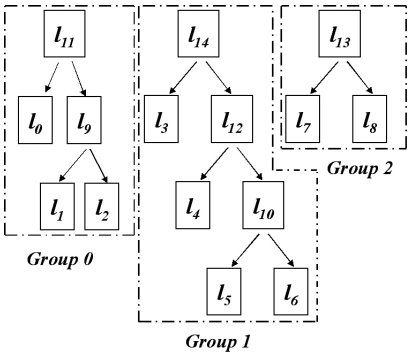

Figure 5: Organisation of Groups in F^r.

$l_{original}$	gr
l_0	0
l_1	0
l_2	0
l_3	1
l_4	1
l_5	1
l_6	1
l_7	2
l_8	2

l_{old1}	l_{old2}	l_{new}	gr
l_1	l_2	l_9	0
l_5	l_6	l_{10}	1
l_0	l_9	l_{11}	0
l_4	l_{10}	l_{12}	1
l_7	l_8	l_{13}	2
l_3	l_{12}	l_{14}	1

Figure 6: Information obtained in the preprocess. Every leaf and every collapse information is classified in one group.

3.1 Obtaining the data structure

First of all, it is necessary to determine the number of independent clusters that form the foliage. In this way, the information obtained from the simplification process FSA has to be processed. This simplification method provides the sequence of leaf-collapse operations to obtain F_{n-1} from the original model F_0. In order to build the multiresolution model, this sequence is processed in a previous process to obtain the number of the group every collapse concerns. It allows us to determine the number of final clusters in the foliage. When processing this information, every leaf in the foliage is assigned to one of the clusters. Regarding the example of Figure 4, the obtained data are shown in Figure 6.

Finally, this information is used to build the multiresolution data structure F^r.

3.2 Data Structure

All the leaves that form the foliage (not only the original ones but also those obtained in the simplification process) are stored in F^r. They are organised as an array of clusters, where all the leaves that form each group are stored together with some information enabling a

Init_group		0		0		0	
Active_leaves		3		4		2	
Changes_number		0		0		0	
leaf_number	next						
		l_1	1	l_5	1	l_7	1
		l_2	2	l_6	2	l_8	2
		l_0	3	l_4	4	l_{13}	-1
		l_9	4	l_{10}	5		
		l_{11}	-1	l_3	3		
				l_{12}	6		
				l_{14}	-1		

Figure 7: Example of data stored in the array of clusters that form F^r.

correct visualisation. In each cluster, leaves are stored traversing the binary tree by levels of depth. In Figure 7 is shown the data organization for the example of the Figure 6.

For every leaf in this array, it is stored the index of the leaf, the index of the vertices that form it and the position of the next leaf in the group to be visualised.

In addition, every cluster stores some information needed to obtain the appropriate sequence of leaves to be visualised in the required approximation:

Active_leaves. Stores the number of leaves to be visualised in the current level of detail.

Init_group. This information is used to store the position in the array of the first leaf to be visualised in the current approximation.

Changes_number. It contains the number of leaf collapse or split operations that have to be processed in the group to obtain the new level of detail required.

Data stored in Figure 7 are the ones of the original foliage F_0. All the clusters have as the *init_group* the 0 position. *Active_leaves* stores the number of leaves to visualise in each group, in this case, 3 leaves in the 0 group, and 4 and 2 for the next ones. In the example, the shaded fields represent the leaves in the current approximation. No changes has to be processed, so *changes_number* is initialized to 0.

Finally, it is necessary to store the sequence of leaf collapses obtained by the FSA. As a consequence of the data organization, only the number of the cluster where the simplification happens is necessary. So, following the example of the data structure of the Figure 4, it is stored the data shown in Figure 8. The field *init_position* indicates the position of the last leaf collapse operation performed. Initially, it is situated in the 0 position of the array.

This information allows us to change the detail of the approximation visualized in real-time.

gr	0	1	0	1	2	1

Figure 8: *Changed_group* structure. Information stored in order to achive an appropriate data retrieval.

Storage Cost Let F^r be an arbitrary LoDF representation; this stores the basic elements that compose a tree, that is to say, its vertices and polygons. Let $|V^r|$ and $|L^r|$ be the number of vertices and leaves stored in F^r, and V and L be the initial numbers of vertices and leaves that composed the crown of the tree. Note that in our multiresolution model:

$$|V^r| = V \tag{3}$$

since the simplification algorithm LoDF is based on, FSA, does not add new vertices when performing the leaf-collapse operation. Regarding the number of leaves, let L be the initial number of leaves,

$$L = \frac{V}{4} \tag{4}$$

since each leaf is formed by 4 vertices independent of the ones forming the other leaves. In every leaf-collapse operation, two leaves disappear and a new one is included. In the worst case, when the maximum number of leaf-collapse operations are carried out, $L-1$ new leaves are added to the initial L.

$$|L^r| = L + (L-1) = 2L - 1. \tag{5}$$

Let G the number of groups in the model. This number is conditioned by the number of leaves in the coarsest approximation F_{n-1}. In the worst case, when the model processes the maximum number of leaf collapse, F_{n-1} will be formed by a single leaf, that is to say, one group.

$$G = 1 \tag{6}$$

Let us suppose that the storage cost of an integer, real number or pointer is one word. The current implementation is not optimised for space, so each leaf would therefore have a cost of 6 words and each vertex 3. In this case, the main data structure of the model has a cost of:

$$3|V^r| + 6|L^r| + 3G \approx 3|V^r| + 6|L^r| \tag{7}$$

The number of leaf collapses conditions the size of the *Changed_group* structure. As said above, if we add $L-1$ new leaves, then this will be the number of leaf-collapse operations processed in our model. So, the size of *Changed_group* is $L-1$, i.e., L. Finally, the data structure cost is:

$$3|V^r| + 6|L^r| + L \tag{8}$$

Summarising, applying the equations 4 y 5, we can say that the storage cost of LoDF is O(V).

$$3V + 13V/4 \approx 6,25V \tag{9}$$

3.3 Retrieval Algorithms

The geometric data of the original model F_0 are initially stored in the graphics card.The application where the multiresolution representation of the foliage is used determines a level of detail depending on the relevance of the tree in the final image. The number of leaves that form the approximation can be changed in real time, adapting it to the requirements of the application. Once this number has been determined, two cases can happen:

- If the current number of leaves is greater than the determined by the application, some leaf collapse operations have to be performed. Considering that every collapse operation eliminates two leaves and adds a new one, it will be performed as many simplification operations as the number of leaves that have to be reduced in the new approximation.
- If the current number of leaves is less than the determined by the application, some leaf split operations have to be performed. Then, it will performed as many leaf split operations as the number of leaves have to be increased in the level of detail.

The sequence of simplification operations, as it is said in the previous section, is determined by the groups where each collapse happens (Figure 8). The level of detail of the current approximation can be changed traversing this information. Each time one simplification operation has to be performed in one cluster, it is increased its field *changes_number* in one.

Once this process is finished, all the clusters whose stored field *changes_number* is other than 0, has to be updated. In every affected group, some process has to be carried out.

Reducing detail. Each collapse operation will be as follows:

- *init_group* will be updated with the new position of the first leaf to be visualised,

$$init\_group + 2 \times changes\_number \tag{10}$$

  because in each leaf collapse two leaves disappear.
- the *active_leaves* will be updated with the new number of leaves to be visualised. The new value will be

$$active\_leaves - changes\_number \tag{11}$$

Init_group		4		2		0	
Active_leaves		1		3		2	
Changes_number		2		1		0	
leaf_number	next	l_1	1	l_5	1	l_7	1
		l_2	2	l_6	2	l_8	2
		l_0	3	l_4	4	l_{13}	-1
		l_9	4	l_{10}	5		
		l_{11}	-1	l_3	3		
				l_{12}	6		
				l_{14}	-1		

Figure 9: Data information after processing three leaf-collapse operations from F_0.

Increasing detail. For each split operation, the algorithm will involve the inverse process, that is to say:

- regarding the field *init_group*,

$$init\_group - 2 \times changes\_number \tag{12}$$

- and every split operation involves increasing the number of leaves to be visualised by one, so the field *active_leaves*

$$active\_leaves + changes\_number \tag{13}$$

Let follow the example of data shown in Figure 4 and the sequence of simplification operation shown in Figure 8. In the case of performing three leaf-collapse operations from the data organisation of Figure 7, according to the stored information, these would affect groups 0 and 1. New data structure will be as the one shown in Figure 9. The leaves to visualise in the current approximation are the shaded ones. Besides, the new value stored in *init_position* of the *Changed_group* data structure would be 3. The last process, once the affected groups have been updated, is to reset to 0 the *changes_number* field into all the groups.

4 VISUALISATION PROCESS

Every simplification operation only affects one group. This characteristic allows us to send only information about the group affected to the graphics card, instead of sending all the geometry of the foliage. Once the cluster affected is updated, the new leaves to be visualised will be the only information that will pass to the hardware. The rest of the groups will remain in the graphics card without modifications, that is to say, just as they were before the change. Thus, we will be updating the information in the card group by group. The model also offers the possibility of doing more than one simplification operation at a time and only those groups where some change has taken place will be sent to the hardware. This process reduces the visualisation time of a

tree considerably if we compare it with other methods that do not take advantage of the coherence property of the graphics card.

5 RESULTS

The method developed here was implemented with OpenGL on a PC with Windows XP operating system. The computer was a dual Pentium Xeon at 1.8GHz. with an NVIDIA GeForce 6800 Series GPU. The trees used in our study were modeled by the Xfrog application [22]. Results are shown in Figures 10 and 11.

In the experiments, the level of detail varies between 0 and 1, 0 being the most detailed approximation and 1 the least. In the figures it is shown how the number of leaves changes for each level of detail. The tests we carried out consist in traversing all levels of detail with a uniform *step*. This step represents the number of leaves that will appear or disappear in every approximation. It is proportional to the difference in the number of leaves between F_0 and F_{n-1}.

Results obtained using LoDF are compared with the ones obtained using VDF. Besides how the number of leaves changes, two graphs are offered in every figure, showing the results:

Total time. Time that the models spends on extracting and also visualising each level of detail.

Extraction time. Time that the models use to extract the necessary geometry to change from the most detailed approximation to the coarsest one.

In the figures it is easily seen how the presented model remarkably improves the results obtained with the VDF model. The hardware-oriented design of the structure accelerates the time employed in visualising a level of detail from a 63% in the case of the *Betula lenta* to a 88%, and in the case of the *Taxus baccata*. The extraction time is also reduced. It also depends on the number of leaves that form the foliage. The more leaves the foliage has the higher percentage of saving visualisation and extraction time it has. In the case of the *Betula lenta* this time is reduced in a 42% and in the case of the *Taxus baccata*, in 70%.

In Figure 12, some different levels of detail of the tree Carya ovata are shown. They vary in a uniform way from 114.114 to 28.528 leaves. Besides, the different approximations are composed following the distance to the viewer criterion. It can be observed how the representations maintain a high similarity with the original tree.

6 CONCLUSION AND FUTURE WORK

In this article we have presented a new multiresolution model that exploits the characteristics of current graphics hardware. This model, *Level of Detail Foliage* allows us to change the level of detail of a foliage representation in a continuous way. Its main advantage is that it tries to reduce the traffic of data through the AGP/PCIe bus diminishing the information that is sent to it when a change of level of detail is produced. This is obtained by grouping leaves in independent clusters and only modifying a small set of data.

This model allows us to represent forest scenes in interactive applications by instancig trees. It produces better results than current models based on images or points. Using geometry to visualise foliage makes it possible to render every detail of the tree. Another advantage of the geometry is that this can be adapted to the characteristics of the graphics hardware.

Another line of research we are currently working on is to obtain advanced ilumination effects and animation of the foliage. We are working on taking advantage of graphics hardware programming.

ACKNOWLEDGEMENTS

This work has been supported by the Spanish Ministry of Science and Technology (TIN2004-07451-C03-03 and FIT-350101-2004-15), the European Union (IST-2-004363) and FEDER funds.

REFERENCES

[1] P. Decaudin and F. Neyret. Rendering forest scenes in real-time. In *Rendering Techniques '04 (Eurographics Symposium on Rendering)*, pages 93–102, 2004.

[2] O. Deussen, C. Colditz, M. Stamminger, and G. Drettakis. Interactive visualization of complex plant ecosystems. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 219–226. IEEE Computer Society, 2002.

[3] I. Garcia, M. Sbert, and L. Szirmay-Kalos. Leaf cluster impostors for tree rendering with parallax. In *Proc. Eurographics 2005 (Short Presentations)*. Eurographics, 2005.

[4] G. Gilet, A. Meyer, and F. Neyret. Point-based rendering of trees. In P. Poulin E. Galin, editor, *Eurographics Workshop on Natural Phenomena*, 2005.

[5] D. House, G. Schmidt, S. Arvin, and M. Kitagawa-DeLeon. Visualizing a real forest. *IEEE Computer Graphics and Application*, 18(1):12–15, 1998.

[6] A. Jakulin. Interactive vegetation rendering with slicing and blending. In A. de Sousa and J.C. Torres, editors, *Proc. Eurographics 2000 (Short Presentations)*. Eurographics, 2000.

[7] J. Lluch, E. Camahort, and R. Vivo. An image based multiresolution model for interactive foliage rendering. *Journal of WSCG'04*, 12(3):507–514, 2004.

[8] P. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 95–102. ACM Press, 1995.

[9] D. Marshall, D. Fussell, and A.T. Campbell III. Multiresolution rendering of complex botanical scenes. In Wayne Davis, Marilyn Mantei, and Victor Klassen, editors, *Graphics Interface*, pages 97–104, 1997.

[10] N. Max. Hierarchical rendering of trees from precomputed multi-layer z-buffers. In Xavier Pueyo and Peter Schröder, editors, *Rendering Techniques '96, Proceedings of the Eurographics Workshop*, pages 165–174. Eurographics, Springer-Verlag, 1996.

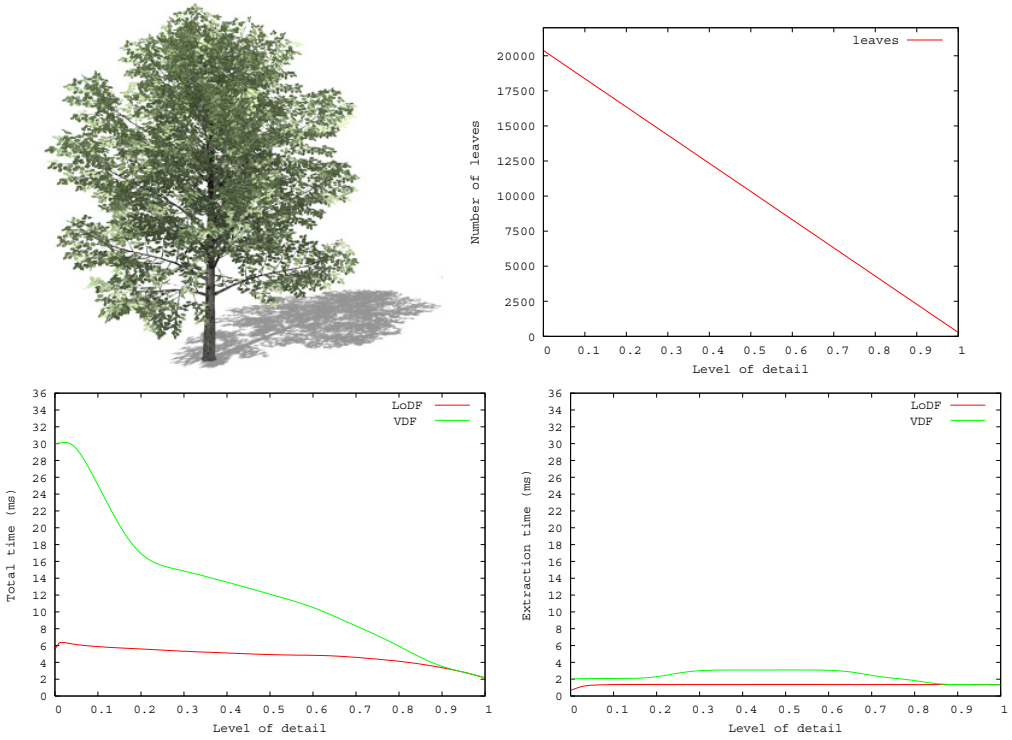

Figure 10: Results obtained for the tree *Betula lenta*.

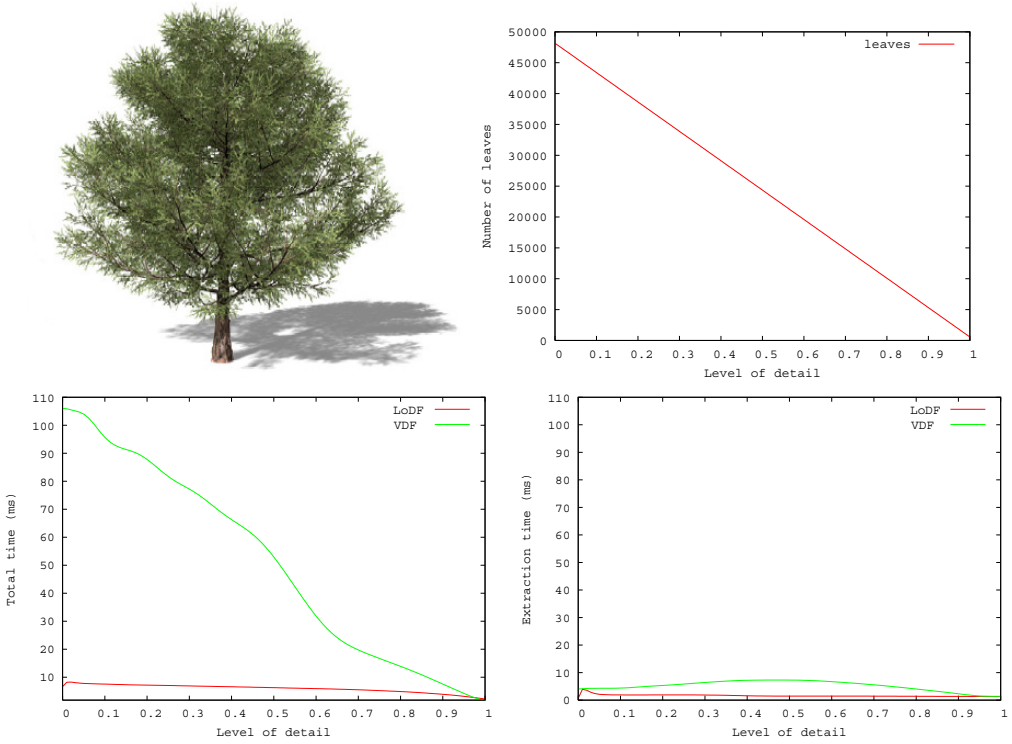

Figure 11: Results obtained for the tree *Taxus baccata*.

Figure 12: Above, from left to right: 114.114 leaves, 85.585 leaves, 57.057 leaves and 28.528 leaves. Below, composition of different aproximations following the distance to the viewer criterion.

[11] A. Meyer and F. Neyret. Interactive volumetric textures. In George Drettakis and Nelson Max, editors, *Eurographics Rendering Workshop 1998*, pages 157–168. Springer Wein, 1998.

[12] A. Meyer, F. Neyret, and P. Poulin. Interactive rendering of trees with shading and shadows. In *Eurographics Workshop on Rendering*. Springer-Verlag, 2001.

[13] A. Reche, I. Martin, and G. Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, 23(3):720–727, 2004.

[14] W. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 313–322. ACM Press, 1985.

[15] I. Remolar, M. Chover, O. Belmonte, J. Ribelles, and C. Rebollo. Geometric simplification of foliage. In I. Navazo and Ph. Slusallek, editors, *Proc. Eurographics 2002 (Short Presentations)*. Eurographics, 2002.

[16] I. Remolar, M. Chover, J. Ribelles, and O. Belmonte. View-dependent multiresolution model for foliage. *Journal of WSCG'03*, 11(2):370–378, 2003.

[17] J. Shade, D. Lischinski, D. H. Salesin, T. DeRose, and J. Snyder. Hierarchical image caching for accelerated walkthroughs of complex environments. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 75–82. ACM Press, 1996.

[18] J. Shade, S.Gortler, L. He, and R. Szeliski. Layered depth images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242. ACM Press, 1998.

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

[20] M. Stamminger and G. Drettakis. Interactive sampling and rendering for complex and procedural geometry. In S.Gortler and C.Myszkowski, editors, *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 151–162. Springer-Verlag, 2001.

[21] J. Weber and J. Penn. Creation and rendering of realistic trees. In Robert Cook, editor, *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 119–128. ACM Press, 1995.

[22] Greenworks: Organic software. `http://www.greenworks.de/`, 2005.