Hardware-Oriented Visualisation of Trees

C. Rebollo, I. Remolar, M. Chover, and J. Gumbau

Departamento Lenguajes y Sistemas Informáticos
Universitat Jaume I, 12071 Castellón, Spain
{rebollo, remolar, chover, gumbau}@uji.es

Abstract. Real-time rendering of vegetation is currently a problem in need of a solution. The large number of polygons that form this kind of objects means that current hardware cannot achieve interactive rendering of outdoor scenes. This paper deals with the problem and it presents a multiresolution scheme that allows us to represent the whole geometry of the trees using both uniform and variable levels of detail. The method presented here models the trees using two multiresolution models. This is due to the different characteristics of the geometry that forms them. The trunk is modelled by *LodStrips*, a model oriented towards representing continuous meshes, and the foliage is modelled by the multiresolution model *Level of Detail Foliage*, presented in a previous work. In this paper, it has been efficiently implemented and extended to allow us to change the level of detail in a variable way, by adapting the resolution of this part of the tree to certain criteria determined by the application. Both of them have been designed to be hardware-oriented. They take advantage of the graphics hardware by adapting the data structures and the rendering algorithms to make the visualisation time efficient. Finally, the multiresolution scheme presented in this paper is compared with the only work that has appeared up to now that uses the same technique.

1 Introduction

Most interactive applications currently available are set in outdoor scenes. In these environments, trees and plants make the scenes more similar to the real world. A vast number of works on modelling vegetal species have appeared up to now, nowadays it is therefore possible to obtain very realistic tree models. However, the problem arises when real-time rendering is required by the application. The more realistic the vegetation objects are, the greater the number of polygons they contain.

Depending on the technique used to solve this problem, methods can be divided into two important groups: image-based and geometry-based rendering. Geometric representation has many advantages, the most important of which is that trees do not lose realism even when the camera is extremely near the object. Another important advantage is that geometry can be stored either in the main memory or directly in the graphics card, thus taking advantage of current graphics-hardware. Moreover, using geometry to represent objects makes it

A. Levi et al. (Eds.): ISCIS 2006, LNCS 4263, pp. 374–383, 2006.

possible to obtain shadows and different lighting effects, as well as greater acceleration in rendering. Geometry-based approaches have used several techniques to achieve interactivity, such as replacing the basic display primitive triangle by points and lines, or using multiresolution modelling techniques.

Multiresolution geometric models have proved to be a good solution for visualising objects made up of a vast number of polygons in interactive applications. These models adapt the geometric detail of the objects to the capacity of present-day graphics systems. Using this technique, objects are represented by means of multiple resolutions, with varying complexities, called levels-of-detail (LoDs). The application can visualise the object using the most suitable LoD and thus avoid, for instance, wasting time on visualising imperceptible details. These models provide two possible solutions: levels of detail of uniform resolution and levels of detail of variable resolution. Multiresolution models do not work properly with the representation of trees because of the characteristics of their geometry [1]: the trunks are modelled by continuous meshes and a set of isolated polygons is used for the foliage.

This paper presents a new hardware-oriented multiresolution approach to represent the geometry of these vegetal species. Two different multiresolution models are used to represent the tree objects: *LodStrips* [2] for the trunk and branches, and *Level of Detail Foliage* LoDF [3] for the foliage. Both of them have been designed within a hardware-oriented approach, thus considerably reducing the visualisation time of the LoDs that are required. In order to adapt the resolution of the different parts of the foliage depending on certain criteria, the data structures and retrieval algorithms of LoDF has been both extended. Finally, these models are compared with the ones used in [4] to represent trees, the results showing how this approach reduces the extraction and visualisation time even if hardware storage is not applied.

After reviewing previous work in section 2, the multiresolution model used to represent the trunk and branches is set out in section 3. Next, section 4 includes an analysis of the one used to represent the foliage. The data structure designed for this model and the retrieval algorithms for a uniform and a variable level of detail of the foliage are presented. Section 5 analyses the results of comparing this new method against the one used in [4] and, finally, in section 6 some ideas for future research are discussed.

2 Related Work

Research into real-time visualisation of detailed vegetal species is aimed at adapting the number of polygons used to represent those plants to the requirements of graphics hardware. As it was said in the previous section, research into interactive visualisation of vegetal species can be grouped into two broad directions: work that uses images or work that uses only geometry to represent the plants.

Image-based rendering. This is one of the commonest methods to represent trees because of its simplicity. Impostors are the most popular example of image-based rendering. This method replaces the geometry of the object with

an image of it textured on a polygon immersed in the scene. Nevertheless, it presents some disadvantages, such as, for example, the loss of realism when the object is close to the viewer. Max [5] adds depth information to the precalculated images. This information allows them to recalculate different views from the stored images of the scene. Other authors obtain 2D images from volumetric textures and combine them depending on the position of the camera, [6]. Some authors, such as Remolar et al. [7], divide the scene into zones depending on the distance from the object to the viewer. Objects farther away from the camera are represented by an image and objects near the viewer are depicted by geometry. García et al. [8] solve the parallax problem by using textures that group sets of leaves.

Geometry-based rendering. This approach does not lose realism when the viewer moves towards the model, but 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 for points or lines [9]. Works such as the ones presented by [1] [10] allow us to interactively adapt the number of points depending on the importance of the object in the final rendered image.

In recent years, several works based on multiresolution models have appeared. Meyer et al. [11] and Lluch et al. [12] use a representation based on multiresolution models of images. The only method so far that works with multiresolution representation of the tree based exclusively on polygons is [7], which the authors have called *VDF*. This work is focused on tree foliage. It adapts the number of leaves that form the foliage in real time.

3 Trunk Representation

The trunk is represented by a continuous general mesh, where the polygons are connected to each other. In our scheme, it is modelling using *LodStrips*. This continuous multiresolution method noticeably improves on previous models, in terms of storage and visualisation cost. The model is entirely based on optimised hardware primitives, triangle strips, and manages the level of detail by performing fast strip updating operations. In addition, it uses stripification techniques oriented towards exploiting vertex buffering.

4 Foliage Representation

Foliage is formed by isolated polygons that represent the leaves. In the presented scheme, it is represented using the multiresolution model LoDF.

In general, a multiresolution representation is constructed from two main elements: the original geometry of the object and the different approximations given by a simplification method. LoDF is based on the *Foliage Simplification Algorithm* (FSA) [13], which provides the coarsest approximation F_{n-1} from the original foliage F_0. The basic simplification operation of this algorithm is *leaf collapse*: two leaves are collapsed into a new one. This operation conditions a

hierarchical structure based on trees. The sequence of leaf-collapse operations is processed to build the new multiresolution representation F^r (Figure 1 left). The LoDF data organisation is 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 the example shown in Figure 1, F_0 is formed by 9 leaves, and F_{n-1} is made up of 3 leaves.

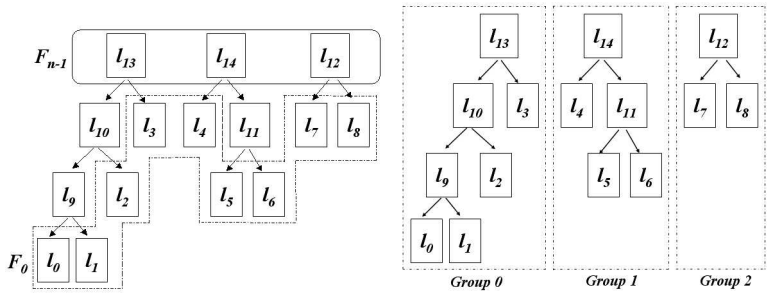

Fig. 1. Example of an F^r structure and the data stored for clusters 0, 1 and 2

In the multiresolution model the inverse operation has been performed in order to increase the level of detail, the refinement operation has been called *leaf split*: one leaf is divided into the two leaves that formed it.

The main characteristic of F^r is that the foliage is divided into independent clusters. As each simplification operation creates a new leaf, each leaf can only belong to one binary tree. Thus, every set of leaves in a binary tree of the data organisation forms one cluster. Following with the previous example, Figure 1 (right) shows the three clusters representing it. These groups determine the data structure used in our model.

4.1 Data Structure of the Foliage

The basic data structure of the LoDF model is an array of clusters (Figure 2). Each cluster stores all the leaves that form a binary tree of the data organisation F^r and some additional information.

In order to make it possible to perform the leaf collapse or split operations quickly, leaves l_i are stored in the clusters following the order of simplification established by the FSA, i.e. by levels of proof in the binary tree. Furthermore, leaves in each cluster are linked following the visualisation order: first of all the leaves that form the original representation are linked and then the ones obtained in the simplification process. These links also follow the order of creation determined by the FSA. This step makes it possible to extract the leaves that form the current LoD quickly. In Figure 2 the link of each leaf in the cluster is represented by n_j, j being the position of the leaf in the array.

In addition, some necessary data for leaves visualisation are also stored in each cluster. These data are:

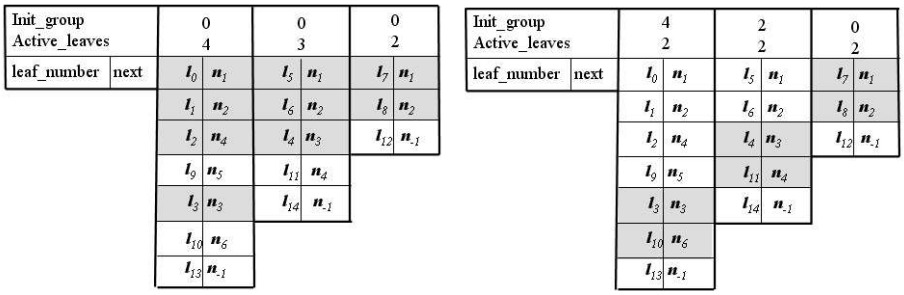

Fig. 2. Example of data stored in clusters 0, 1 and 2 for the most detailed representation (left) and the data stored after performing three leaf collapse operations (right)

- *Init_group*: the position in the array of leaves where the first leaf of the current LoD is stored.
- *Active_leaves*: number of leaves to be visualised in the current approximation.

In order to extract the sequence of leaves that form the required LoD, we must start traversing the group from the position indicated in the *Init_group*. The number of links that the algorithm must consider is the value stored in *Active_leaves*.

In addition to this structure, the multiresolution model also needs to store the numbers of the clusters where every simplification operation that is processed to obtain F_{n-1} from F_0 is performed. This data structure is called *Changed_groups*. Following the example of the data shown in Figure 2, the data stored in this structure are represented in Figure 3.

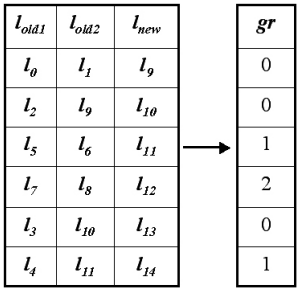

Fig. 3. Left: Data obtained from the FSA. Right: Data stored in *Changed_groups*.

The LoDF data organisation is well adapted to graphics hardware. The geometric data of the model is initially stored in the graphics card. When a change in the LoD is required by the application, the affected clusters are updated. The only information that will be sent to the hardware are the leaves of these updated clusters. The rest of the groups will remain in the graphics card without undergoing any kind of modification, that is to say, just as they were before the

change of LoD. Visualisation time is considerably reduced due to the fact that only the information about the updated clusters is sent to the graphics card instead of sending all the geometry of the foliage.

4.2 Rendering Algorithms for a Uniform LoD

In order to obtain a uniform resolution in the foliage, the leaf collapse operations have to be performed in the order established by the FSA. This sequence has been stored in the array *Changed_groups*. Once the application where the foliage is included establishes the number of leaves for the appropriate LoD, the extraction algorithm determines the number of leaf split or collapse operations that have to be performed in the foliage. Then, a pointer crosses the *Changed_groups* array and updates the *Init_group* and the *Active_leaves* fields in each cluster where an operation has to be performed.

Due to the storing order of the leaves in the clusters, a leaf-collapse is simply performed by increasing the *Init_group* by two units, and a leaf-split by decreasing this field by two units. These operations allow us to automatically update the position where the extraction algorithm starts traversing the leaves of the group. Figure 2 right shows how three leaf collapses affect the initial data, situated on the left of the figure. According to the information stored in *Changed_groups*, two of them are performed in the 0 and one in the 1 clusters. The leaves to be visualised are shaded.

Once the necessary operations have been processed, the visualisation process begins. This algorithm starts traversing the group from the position indicated in the *Init_group*. It follows the links stored in the leaves to achieve the leaf visualisation sequence. The number of links that the algorithm has to consider is stored in *Active_leaves*.

4.3 Rendering Algorithms for a Variable LoD

The main advantage of data organisation is that different clusters can be visualised at different resolutions. This fact makes it possible to represent the foliage with different detailed zones coexisting in the same representation. The application assigns a certain number of leaves to represent the foliage of a tree and the leaves that are visualised can be distributed following some specific criteria.

To retrieve a variable resolution LoD, a criterion or a set of criteria is needed. These criteria decide which part of the object is to be simplified and which part is to be refined. They depend on the final application where the multiresolution object is to be included. In the case of LoDF, a criterion has been implemented in order to determine the most detailed zone. This criterion is the *distance to the viewer*. Clusters situated near the camera require more detail than those farther away from it. The function *MostDetailedZone* calculates the appropriate number of leaves in the cluster depending on this criterion. This function evaluates each cluster and returns the appropriate number of leaves in each one following a linear function. Figure 4 shows two trees where part of their foliage is less detailed than the rest.

The number of leaves to visualise allows us to know the number of leaf collapse or split operations that have to be performed in the clusters: if the number of leaves to be visualised in the required LoD is higher than the number of leaves currently being visualised, then some leaf split operations have to be performed. In the other case, if the new LoD requires less leaves than the one currently being visualised, the algorithm has to perform the appropriate number of leaf collapses. This information is used to obtain the *Init_position* in the leaf array of the cluster. The number of links to be traversed is the number of leaves that we want to visualise, which is returned by the function *MostDetailedZone* and stored in the *Active_leaves* field of the cluster.

Nevertheless, more criteria can easily be added to LoDF if the application makes it necessary to do so.

Fig. 4. Example of variable resolution according to the implemented criterion. A part of the tree is more detailed than other.

5 Results

All the tests used here were implemented in C++ with OpenGL as the graphics library. The trees used in our study was modelled with the Xfrog application [14]. The trunks are made up of 34.202 and the foliages consist of another 95.679.

The method is compared with the one presented by Remolar et al. [7], that is, *VDF+MOM*. In order to achieve a better comparison, the tests designed for use here were performed on a computer with the same characteristics as the one used in that work.

The multiresolution methods *VDF+MOM* are not hardware-oriented, so first of all they were compared with the methods presented in this paper, *LoDF+LodStrip* but without taking advantage of the graphics hardware. In other words, all the geometric data about the *LoDF+LodStrip* models are stored in the main memory in the same conditions as the data in *VDF+MOM*. Two tests were designed in this case: one for uniform changes in the LoD and other one for variable changes. Results are shown in Figure 5.

The tests we carried out consist in traversing different LoDs. In all of them, the LoD varies between 0 and 1, 0 being the most detailed approximation and

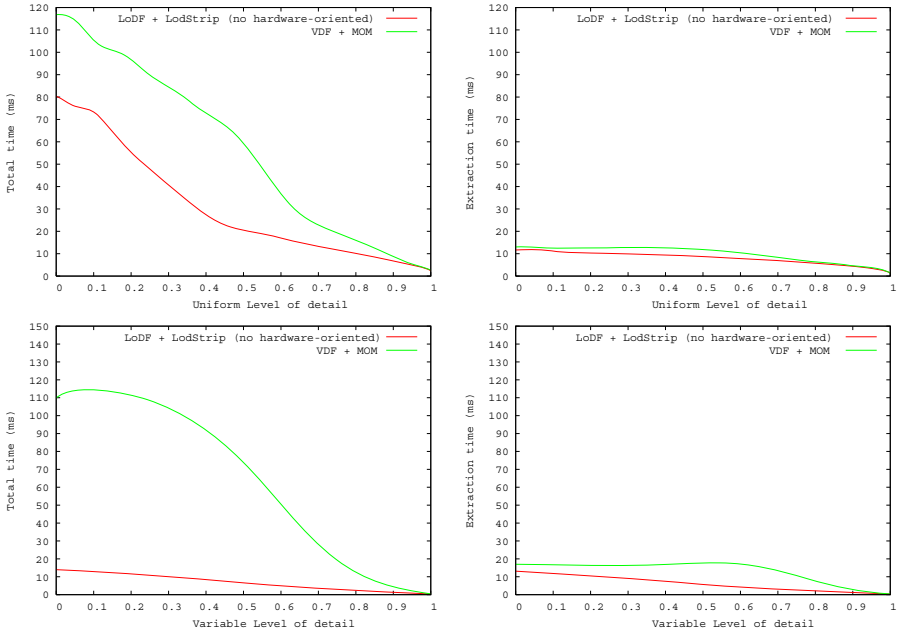

Fig. 5. Results obtained on comparing *VDF+MOM* with *LoDF+Lodstrip*

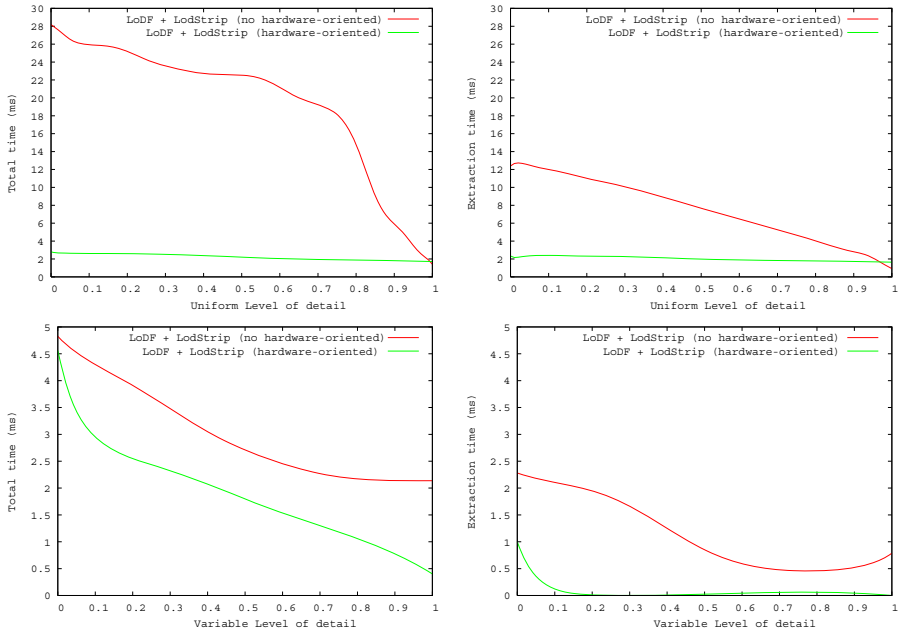

Fig. 6. Results obtained on comparing *LoDF+LodStrip* with and without taking advantage of the graphics hardware

1 the least. The tests were designed following the criterion defined in the multiresolution models. Two graphs are offered in every figure to show the results:

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

Extraction time. Time that the model uses to extract the geometry required to change from one approximation to other one.

In the figures it can be seen that our method reduces the total time considerably. Using *LoDF+LodStrip* trees can be visualised quite a lot faster than by using *VDF+MOM*. Finally, the main advantage of representing a tree using the methods *LoDF+LodStrip* methods is that multiresolution models are graphics hardware oriented. Other tests have been carried out in order to evaluate this characteristic. The same experiments were conducted on a computer with an NVIDIA GeForce 6800 Series GPU. In this case, Figure 6 shows how the total and extraction time for visualising a tree can be reduced by using this property.

A video showing the visualisation of a forest can be obtained from the web page http://graficos.uji.es/rebollo/demo.html.

6 Conclusions and Future Work

In this article we have presented a new multiresolution approach that exploits the characteristics of current graphics hardware. The models *Level of Detail Foliage* LoDF and *LodStrip* allow us to change the level of detail of a tree representation in a continuous way. The main advantage is that they reduce the traffic of data through the AGP/PCIe bus by diminishing the amount of information that is sent to it when a change in level of detail is produced. In the case of LoDF, this is obtained by grouping leaves in independent clusters and only modifying a small set of data.

Both methods are combined in the multiresolution scheme that is presented in order to make it feasible to represent scenes of forest in interactive applications. It allows us to render every detail of the tree even when the viewer is extremely close to it. Because it is adapted to the graphics hardware, it produces better results than current models based on images or points.

One line of research we are currently working on is to obtain advanced illumination effects and animation of the foliage. One of our aims in these studies is to continue to take advantage of graphics hardware programming.

Acknowledgements

This work was 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. Deussen, O., Colditz, C., Stamminger, M., Drettakis, G.: Interactive visualization of complex plant ecosystems. In: VIS '02: Proceedings of the conference on Visualization '02, IEEE Computer Society (2002) 219–226
2. Ramos, J.F., Chover, M.: Lodstrips: Level of detail strips. In: International Conference on Computational Science. (2004) 107–114
3. Rebollo, C., Remolar, I., Chover, M., Ripollés, O.: An efficient continuous level of detail model for foliage. In: Proc. of 14-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2006), UNION agency (2006) 335–342
4. Remolar, I., Rebollo, C., Chover, M., Ribelles, J.: Real time tree rendering. In: Lecture Notes in Computational Science 3039. (2004) 173–180
5. Max, N.: Hierarchical rendering of trees from precomputed multi-layer z-buffers. In Pueyo, X., Schrder, P., eds.: Rendering Techniques '96, Proceedings of the Eurographics Workshop, Eurographics, Springer-Verlag (1996) 165–174
6. Reche, A., Martin, I., Drettakis, G.: Volumetric reconstruction and interactive rendering of trees from photographs. ACM Transactions on Graphics (SIGGRAPH Conference Proceedings) **23**(3) (2004) 720–727
7. Remolar, I., Chover, M., Ribelles, J., Belmonte, O.: View-dependent multiresolution model for foliage. Journal of WSCG'03 **11**(2) (2003) 370–378
8. García, I., Sbert, M., Szirmay-Kalos, L.: Leaf cluster impostors for tree rendering with parallax. In: Proc. Eurographics 2005 (Short Presentations), Eurographics (2005)
9. Stamminger, M., Drettakis, G.: Interactive sampling and rendering for complex and procedural geometry. In S.Gortler, C.Myszkowski, eds.: Proceedings of the 12th Eurographics Workshop on Rendering Techniques, Springer-Verlag (2001) 151–162
10. Gilet, G., Meyer, A., Neyret, F.: Point-based rendering of trees. In E. Galin, P.P., ed.: Eurographics Workshop on Natural Phenomena. (2005)
11. Meyer, A., Neyret, F., Poulin, P.: Interactive rendering of trees with shading and shadows. In: Eurographics Workshop on Rendering, Springer-Verlag (2001)
12. Lluch, J., Camahort, E., Vivo, R.: An image based multiresolution model for interactive foliage rendering. Journal of WSCG'04 **12**(3) (2004) 507–514
13. Remolar, I., Chover, M., Belmonte, O., Ribelles, J., Rebollo, C.: Geometric simplification of foliage. In Navazo, I., Slusallek, P., eds.: Proc. Eurographics 2002 (Short Presentations), Eurographics (2002)
14. Xfrog: Greenworks: Organic software. `http://www.greenworks.de/` (2005)