View-Dependent Multiresolution Model for Foliage

I. Remolar M. Chover J. Ribelles O. Belmonte
Dep. Lenguajes y Sistemas Informáticos
Universitat Jaume I
Campus Riu Sec
Spain 12071, Castellón
{remolar, chover, ribelles, belfern}@uji.es

ABSTRACT

Real-time rendering of vegetation is one of the most important challenges in outdoor scenes. This is due to the vast amount of polygons that are used to model vegetable species. Multiresolution modeling has been successfully presented as a solution to the problem of efficient manipulation of highly detailed polygonal surfaces. In order to construct a multiresolution model, a simplification method must be used. The previously introduced Foliage Simplification Algorithm, FSA, obtains different approximations of a tree model. This automatic simplification method diminishes the number of polygons that form the crown while maintaining its leafy appearance. In this paper a multiresolution model for trees based on this simplification algorithm is presented. Its distinctive property is that the unit of information managed by this scheme is the leaf, four vertices determining two triangles. This characteristic allows us an efficient manipulation of the results obtained by FSA. Our multiresolution representation provides a wide, virtually continuous, range of different approximations that represent the original tree. The main property of this scheme is that trees can be represented with variable resolution: some regions in more detail while the rest is represented in less detail. Here we present the data structures and the traversing algorithms, which favor the extraction of an appropriate level of detail for rendering.

Keywords

Tree Rendering, View Dependent Visualization, Multiresolution Modeling, Level of Detail, Real-Time Rendering.

1. INTRODUCTION

Many of the current interactive applications such as flight simulators, virtual reality environments or computer games take place in outdoor scenes. One of the essential components in these scenes is the vegetation. The lack of trees and plants can detract from their realism. Tree modeling has been widely investigated [Prus90] [Lint99], and its representation has become very realistic. However, tree models are formed by such a vast number of polygons that real-time visualization of scenes with trees is practically impossible.

Real-time visualization of vegetable species has not been extensively explored. Although there are more complex techniques, most of the applications make use of image-based rendering approximations [Max96] [Scha98]. However, geometry is necessary in others. In this case, the most popular method of representation is to use polygonal models. The mathematical simplicity of this type of representation makes it possible to render a great number of polygons with the current graphics hardware. However, due to the vast amount of polygons that compose the tree models, it is necessary to use some method that diminishes the number of polygons that form the object, without loss of the appearance. The most used techniques are occlusion culling or level of detail (LoD) rendering.

In a previous work we presented the *Foliage Simplification Algorithm, FSA* [Remo02], an automatic method of generating different levels of detail of a same tree without losing similarity with the original model. It is the only automatic simplification algorithm of foliage that diminishes the number of polygons that form the crown while maintaining its leafy appearance. The key to the algorithm is leaf collapse.

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

In this paper a new continuous multiresolution model is presented that allows us an efficient manipulation of the results obtained by FSA. The View Dependent Multiresolution Model for the Foliage *VDF* can perform the interactive visualization of the trees. The application can make a selection of the LoD in running time in order to establish a balance between the number of polygons with which the object will be represented and the amount of time needed to visualize it. In this paper, the data structure and the extraction algorithm of the required level of detail is presented.

VDF is the only scheme specially designed for the set of isolated polygons that form the foliage in a tree. The multiresolution models that have appeared up to now deal with general meshes and do not work properly with this kind of mesh. One of the main properties of our multiresolution model is that the trees can be represented with variable resolution, thus allowing different LoDs to coexist in the crown.

In accordance with [Ribe02], our model would be classified within the Hierarchical Representation group. The schemes in this group can manage view-dependent approximations of the objects, but none of them can handle the information obtained from the FSA. This is because the handled information in the appeared multiresolution schemes are the vertices, and in VDPM are the leaves. [Hopp97] [Xia96] [ElSa99] [ElSa99b] use methods based on the union of pairs of vertices in the construction process simplification. [Lueb97] uses a method based on vertex clustering-- a set of vertices is collapsed into one vertex. Our model manages leaves each formed by four vertices.

After reviewing previous work in section 2, section 3 briefly reviews the Foliage Simplification Algorithm. The multiresolution model specially designed for this kind of 3D models is studied in depth in section 4. In section 5, the visualization algorithm is analyzed. This section shows how we can visualize both variable levels of detail and uniform ones. Section 6 presents the results and in section 7 they are analyzed. Some ideas for future research are discussed in this section.

2. RELATED WORK

The research conducted about vegetation could be divided into two broad fields: the generation of plants and trees, and their visualization. Vegetation modeling has been extensively explored. The most important works in this field are Lindermayer-systems [Lind68] [Prus90], used for generating realistic models of trees. Other solutions combine grammar-based modeling with traditional techniques [Lint99]. Apart from the great number of studies that have appeared in the literature, some commercial applications have been developed for modeling trees, such as Xfrog (http://www.greenworks.de) (Figure 1), OnyxTree (http://www.onyxtree.com) and AMAP (http://www.bionatics.com).

Figure 1: Tree modeled with the commercial modeling tool Xfrog: 89.533 triangles.

In contrast, real-time vegetation rendering continues to be a problem at present. Image-based rendering is one of the most widely used techniques due to its simplicity. Different works have been presented in this field, such as billboarding [Max96], Layered Depth Images [Scha98], multi Z-buffers [Max96], etc. [Mars97] proposes a representation of botanical scenes by integrating polygonal representations of large objects with tetrahedron approximations of the less representative parts of the scene. Other solutions have been proposed in the literature but very few of them work with geometry.

Level of Detail rendering tries to reduce the complexity of polygonal data sets in a smart manner. A discrete set of levels of detail can be constructed with several independent representations of the same tree with different approximations. Simplified versions of these objects can be obtained, in the case of using L-systems, by limiting the number of polygons at the time of generating the object.

As regards the continuous level of detail, the multiresolution models that have appeared up to now can be classified in three main groups, according to their representation [Ribe02].

- Multi-triangulation [Flor97] [Pupp98] is a structure that constitutes a general framework for continuous multiresolution schemes. In spite of these characteristics, the implementation presented in [Flor98] is based on a vertex decimation method.
- Incremental representations [Hopp96] [Pupo97] [Hopp98] were mainly developed for application

in progressive transmission and interactive visualization using approximations of uniform level of detail.

- Hierarchical representations [Xia96] [Hopp97] [Lueb97] [ElSa99] [ElSa99b] are mainly used to create new approximations formed by several levels of detail which coexist in different areas of the surface.

The multiresolution models in the literature deal with general meshes. VDF is specially designed for the isolated polygons that represent the foliage. It could be classified in the last group because it allows view dependent visualization. All of the multiresolution schemes in this group manage the vertices in the mesh but the unit of information managed by VDF is the *leaf*, four vertices representing two triangles.

3. REVIEW OF FOLIAGE SIMPLIFICATION ALGORITHM

The method of simplification *Foliage Simplification Algorithm* [Remo02] diminishes the number of polygons that form the crown while maintaining its leafy appearance (Figure 8). The key to the algorithm is leaf collapse. Two leaves are transformed into a single one, so that the area of the new leaf is similar to the area formed by the two leaves initially. An error function is used to determine which pair of leaves will be simplified to create a new one.

From the list of the active leaves, the algorithm repeatedly selects a pair that minimizes the error function. These two leaves disappear and a new one is obtained. The collapsed leaves are eliminated from the list of actives and then the new leaf is evaluated with the leaves that remain in the foliage.

The simplification method is characterized by two elements: the measurement that specifies the cost of collapsing two leaves, and the position of the vertices that form the newly created leaf.

Leaf Collapse Cost.

Given a set of candidate leaves to be collapsed, a pair will be chosen so that the error function is diminished. This function combines distance and planarity between the pair of evaluated leaves.

Vertex placement.

The simplification algorithm does not introduce new vertices in the model. The vertices of the new leaf will be two vertices of each of the collapsed leaves. This method will allow us to maintain an area similar to that of the two original leaves.

4. MULTIRESOLUTION MODEL OF THE FOLIAGE

The trees used in our study are modeled by the Xfrog application [Lint99]. They are very realistic, but are generally made up of more than 50 000 polygons each. This is a disadvantage when it comes to generating images in an interactive way.

Trees can be separated into two different parts:

- the solid component of the tree, the trunk and the branches, represented by triangle meshes.
- and the sparse component, the foliage or leaves, formed by a set of independent polygons.

The multiresolution models that have appeared up to now have been defined for general meshes, not for isolated polygons. In this way, the trunk and branches can be modeled by any of the existing schemes, but they do not work properly with the sparse component of the tree. The model presented in this paper, VDF, allows us to represent the foliage with different levels of detail.

Our multiresolution model is created from a sequence of leave collapses determined off-line by *FSA*. The basic data structure defined for the model is a binary tree. Each collapsed pair of leaves conditions a hierarchic relation. The node representing the new leaf is the father of the nodes representing the collapsed leaves. The data structure is created bottom-up as a binary tree.

The leaf nodes of this structure are the polygons representing the leaves of the tree with a maximum degree of detail, and the root nodes are the polygons needed to represent the foliage with a minimum amount of detail. In this way, the structure is made up of a "forest" of binary trees. Figure 2 shows an example of the structure. Let L_0 be the set of leaves forming the original foliage, in this case only 9 leaves. The root nodes correspond to the worst level of detail, L_n. Eventually, this level of detail would be represented by only 3 leaves.

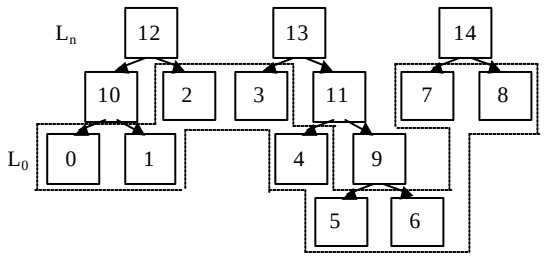


Figure 2: Example of data structure of the multiresolution model of the foliage of a tree.

In order to define a multiresolution model, it is necessary to establish the data structures to store the

information and the algorithms for accessing these structures in order to retrieve this information in the most efficient manner. These algorithms select the most appropriate level of detail in real time and following a criterion, such as distance from the observer or importance of the object in the scene.

Basic Data Structure

The main data structures are shown in Figure 3.

All the geometry of the foliage is stored in *Foliage*. The main function of this structure is to store a list of all the vertices and leaves that compose the model. A list of the leaves that are visualized in a determined LoD, *Actl*, is also stored. To achieve a better management of this list, it stores pointers to the first and last one of the elements that compose it.

```
struct Foliage{
   Vertex  *Vertices;
   Leaf    *Leaves;
   Active  *Actl;
   int      beg,end;
}
struct Leaf{
   int    leafVertex[4];
   leaf*  father;
   leaf*  left;
   leaf*  right;
}
struct Active{
   int  index;
   int  prev;
   int  next;
}
```

Figure 3: Main C++ data structures

The structure *Leaf* stores the vertices that make up a leaf, as well as the pointers necessary to maintain the tree structure: a pointer to the parent leaf, and two pointers to the two children leaves.

Finally, the structure *Active* maintains the fields necessary to codify a doubly connected list of the leaves that are visualizing in a certain LoD.

4.1.1 Storage cost

Let F_r be an arbitrary VDF representation; this stores the basic elements that compose a tree, that are, its vertices and polygons.

Let $|V_r|$ and $|L_r|$ be the number of vertices and leaves stored in F_r, and V y L be the initial numbers of vertices and leaves that composed the crown of the tree. Note that in our multiresolution model:

$$|V_r| = V$$

as it does not add new vertices when performing the leaf-collapse operation. Regarding the number of leaves, let L be the initial number of leaves,

$$L = V / 4,$$

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 this case L-1 new leaves are added to the initial L .

$$L_r = L + (L\text{-}1) = 2L\text{-}1.$$

Let us suppose that the storage cost of an integer, real or pointer is one word. The current implementation is not optimised for space, so each leaf would therefore have a cost of 7 words and each vertex of 3. In this case:

$$3|V_r| + 7|L_r| = 3V + 7(2L\text{-}1) \cong 3V + 7V/2 \cong 6V$$

Summarizing, we can say that the storage cost of VDF is O(V).

5. VISUALISATION OF FOLIAGE

The main algorithm defined for accessing these structures uses a list of active leaves L_s that enables us to know which leaves are visualized in the current level of detail. The basic idea consists in checking L_s, verifying that every leaf in this set is valid for the required level of detail. If this is not true, it is necessary to apply one of the following operations (Figure 4):

- *collapse*. The leaves are drawn in the worst detailed zone and they have to be collapsed.
- *split*. The leaf is not valid, because it is in the most detailed zone. It has to be refined.

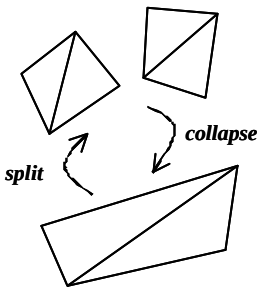


Figure 4: Operations that can be performed in this model.

VDF has been designed in order to achieve a view dependent visualization. Despite the fact that it is possible to obtain different uniform levels of detail.

These will be used when the tree is outside a certain zone of interest and it is not necessary to visualize it with much detail. We will now see an analysis of each one of the cases.

Variable Levels of Detail

5.1.1 Selective refinement algorithm

The validity criteria of the leaves depend on the application. A criterion decides which part of the object is simplified and which part is refined. The user could decide interactively which regions should be displayed with higher or lower detail, or the application could also take the decision. Several authors [Hopp97][Xia97][Lueb97][ElSan99] have defined criteria about the automation of the selection of the areas to be represented in higher detail, such as local illumination, view frustum, screen-space projection, etc. The solutions proposed in the literature can easily be added to our scheme, since our algorithm is independent of the criteria to be used.

The core of the traversal algorithm is summarized below in Figure 5.

```
for each leaf l ∈ Ls
{
 if(not(Criteria(l))){ //COLLAPSE
   if(HaveFather(l)){
    if(not(Criteria(sister)){
      ready = TRUE
      if(not isActive(sister))
        ready = ForceCollapse(sister)
      if ready collapse (l);}
    }
 }
 else { //SPLIT
   nodoSplit = CheckSplit (l);
   if (nodoSplit != NOTFOUND)
      ForceSplit (l,nodoSplit);
 }
}
```

Figure 5: Pseudocode of the extraction algorithm of the required level of detail.

In order to obtain a visualization dependent on the view, the list of active leaves will be checked. If the checked leaf is not in the zone of interest, two requirements will be verified in order to apply the collapse operation. First of all, it is necessary for this leaf not to be a root node. Next, it is verified that the leaf-sister is active and it is not in the zone where maximum detail is required. If all these conditions are fulfilled, these two leaves will not be visualized in the following level of detail and will be replaced by their parent leaf. In our model, it is not necessary to check before performing an operation if this one maintains the continuity in the mesh as it happens in multiresolution models for arbitrary meshes.

On the other hand, if the checked leaf is in the zone where maximum level of detail is required, it will be verified whether it fulfills the conditions to perform the split operation. This active leaf must have children and it is only necessary for one of them to be in the zone of the crown where more detail is desired. If this is so, this leaf will not be active and it will not be visualized in the following level of detail, being replaced by its children.

The *Criteria* function is TRUE if the evaluated leaf belongs to the zone that must have more detail. If it is in the coarsest zone, the preconditions are evaluated in order to perform a collapse transformation. The *ForceCollapse* function is called when the sister of the candidate leaf is also in this zone, but it is not active. In this case, some of its descendants are drawn in the current level of detail. This function checks its descending active leaves and evaluates whether they are in the minimum detail zone. If they are, they will be collapsed. This process is repeated until the sister-leaf is active. If any of its descending leaves are in the zone of interest, the function returns FALSE, indicating that the target collapse cannot be performed.

```
Procedure CheckSplit(l){
 n= NOTFOUND;
 if HaveChild(l)
    if(Criteria(l.left)or
Criteria(l.right)
      n=l.left
    else {
       n=CheckSplit(l.left)
       if (n == NOTFOUND)
         n=CheckSplit(l.right)
    }
 return (n)
}
```

Figure 6: Pseudocode of the CheckSplit procedure.

As regards the operation split, the *CheckSplit* function (Figure 6) deals with the sub-tree formed by the descendants of the checked node *l*. The procedure evaluates its descendants until it finds a node *n* that is in the zone of interest. When this happens, the function *ForceSplit* refines all the

descendants of node *l*, until the activation of this leaf *n* is achieved.

5.1.2 Examples

Figure 7 shows an example of view-dependent visualization.

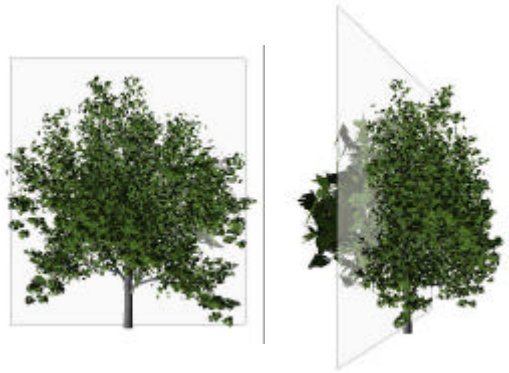


Figure 7: Example of variable resolution in the foliage, which is more detailed in front of the plane.

Figures 11 and 12 show further results of different resolutions in the same crown. The foliage of this tree was initially formed by 20.376 leaves, that is, 40.752 triangles. The validity criteria used in these cases have been:

- a sphere centred in the crown. The leaves inside this sphere are visualized with the minimum level of detail (Figure 11).
- a plane that vertically divides the crown. The foliage in the zone nearest the observer will be visualized with the maximum level of detail, whereas the zone behind the plane has a coarser resolution (Figure 12).

Uniform Levels of Detail

Sometimes a uniform level of detail is required by the application. If the tree is completely outside the zone of interest, this will have to be drawn with the worst level of detail in all its crown. The LoD will increase at the same time the viewer moves towards the tree.

In spite of this, the model has been designed to obtain view dependent visualization and different uniform LoDs can be obtained according to the interest of the tree in the scene. For this reason, special functions have been designed to obtain a representation of foliage with a certain number of leaves.

Should leaves have to be collapsed, this will be a linear process. The model checks the active leaves until it finds the ones that fulfill these conditions:

- not being root of a binary tree structure, and
- its leaf-sister is active as well.

In this case, the collapse operation could be performed. This process will stop when the actual LoD has the number of leaves required by the application or when the minimum level of detail has been achieved.

In the other case, if the application requires a higher level of detail, the process would be similar. Crossing the list of active leaves, it is only necessary to verify that this leaf has children. If this is so, this leaf would be eliminated from the active list and its children would be added.

5.1.3 Examples

Figure 8 shows some examples of uniform resolution in the foliage of a tree.

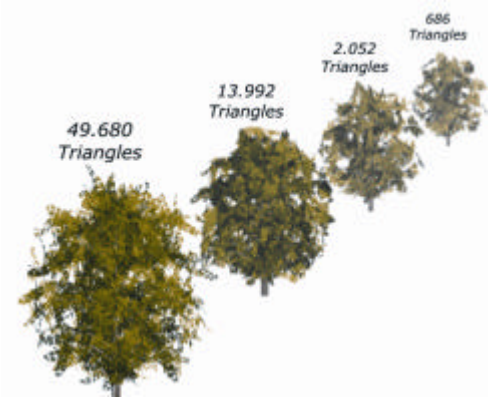


Figure 8: Different simplifications of the foliage in a tree.

Different levels of detail of the tree in Figure 1 are presented in Figure 10.

Time Complexity

In general, the algorithm traverses the active leaf list, performing the split or ecol operations. Let F_A and F_B be two different levels of detail of the same crown, and F_n the coarsest representation. The computational cost for transforming F_A into F_B, is in the worst case $O(F_A+F_B)$. This refinement, $F_A\ F_n\ F_B$ could possibly require $O(F_A)$ ecol and $O(F_B)$ split operations. Considering as constant the cost of performing these operations, the bottleneck of the refinement algorithm is the traversal of the active leaf list.

For continuous view changes, F_B is normally similar to F_A, and the simple traversal of the active vertex list is the bottleneck of the incremental refinement algorithm. Note that the number of active leaves is typically much smaller than the number L of original leaves.

6. RESULTS

The developed method has been implemented with OpenGL on a PC with Windows 2000 operating

system. The computer is a dual Pentium Xeon at 1.8GHz. with a graphical processor NVIDIA Quadro4 with 64MB.

The tree used in our experiments is the one shown in Figure 1 formed by 88.443 polygons. Its foliage was initially formed by 20.376 leaves, that is, 40.752 triangles and its trunk by 47.691 polygons.

The trunk and the branches have been represented using our previous multiresolution model for general meshes *Multiresolution Triangle Strips* [Belm02].

The tests measure the frames per second in a scene where the number of trees are increased. The camera follows a circular path around this scene.

The foliage is rendered using three methods:

- *Geometry*: every tree in the scene is represented with the maximum level of detail.
- *Uniform levels of detail:* The tree nearest the viewer are always represented with full detail. This detail is diminished as the trees are situated far away from the viewer.
- *Variable levels of detail*: The same as the previous test, the tree nearest the viewer are always represented with full detail. The interest zone has been determined by a sphere centred in the foliage. Leaves inside this sphere are represented with the worst detail, and the rest of the foliage with the highest. As the trees are situated far from the viewer, the radius of the sphere is incremented.

Figure 9 shows the results for these test.

As we can see in this chart, multiresolution modelling increases the number of frames visualized per second. This is because the number of polygons that are drawn diminishes extraordinarily without reducing the realism of the scene. In the chart is shown that frame rate improves with the use of view-dependent visualization. This allows us to render a foliage with different level of details, depending on the interest of the tree in the scene.

In figure 13, a scene of our test is shown. For more information and colour images, please visit http://graficos.uji.es/trees.

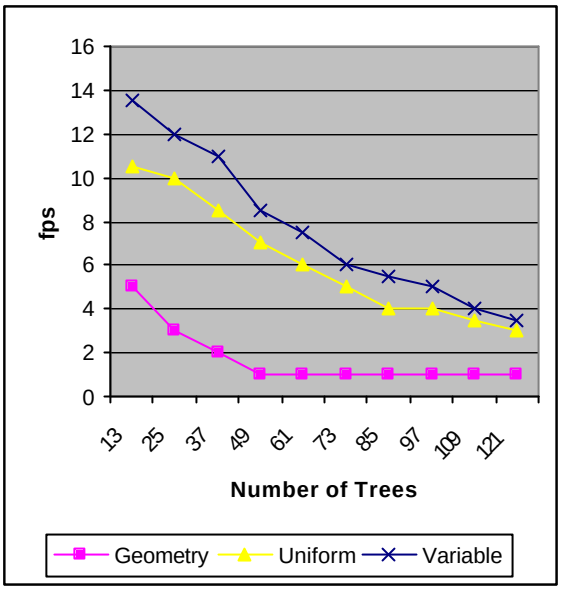


Figure 9: *Results of the experiments*.

7. CONCLUSIONS AND FUTURE WORK

In this paper has been presented a multiresolution model specially designed for the foliage in trees, VDF. This is the only scheme specially designed for the set of isolated polygons that form the foliage in a tree. The multiresolution models that have appeared up to now deals with general meshes and they do not properly work with the crown in a tree. VDPM provides a wide range, virtually continuous, of different approximations that represent the original tree without loss of leafiness The main property of this scheme is that trees can be represented with variable resolution: some regions in more detail while the rest of it is represented in less detail.

Using multiresolution models of the trees allows us to adjust the level of detail to the application requirements. In this way, fewer polygons are used to draw distant objects. Due to the great amount of work that has been carried out in image-based rendering, the next step is to combine the multiresolution modeling with this technique, particularly the dynamic generation of impostors. Drawing a smaller number of polygons will lead to a reduction in the cost of impostor generation. Trees will be rendered using dynamic impostors that take advantage of the frame-to-frame coherence inherent in three-dimensional scenes. Impostors avoid the need to redraw all the geometry of the scene continuously. The remote trees are drawn with an impostor that has been generated with few polygons. When trees are close to the viewer, the front parts will be drawn with geometry and the rear side with impostors.

Another line of research we are currently working on is the improvement of the realistic representation of trees taking illumination into account. We are developing solutions based on the use of light maps. On the other hand, the visualization of scenes with many trees requires the use of techniques such as hierarchical subdivision of scenes [Shad96], occlusion-culling methods and multi-layered impostors [Deco99]-- topics that are all currently being studied.

8. ACKNOWLEDGMENTS

This work was supported by the Spanish Ministry of Science and Technology grant TIC2001-2416-C03-02.

9. REFERENCES

[Belm02] Ó. Belmonte, I. Remolar, J. Ribelles, M. Chover, M. Fernández, “Efficient Implementation of Multiresolution Triangle Strips”, Proc. of the Computational Science 2002 Conference, vol. 2, pp. 111-120, 2002.

[Deco99] X. Decoret, G. Schaufler, F. Sillion, J. Dorsey, “Multi-layered impostors for accelerated rendering”, Eurographics’99, 18(3), 1999.

[Flor97] L. De Floriani, E. Puppo, P. Magillo, “A formal approach to multiresolution hypersurface modeling”, Straber W., Kein R., Rau R., editors, Geometric modeling:theory and practice, Berlin:Springer, 1997.

[Flor98] L. De Floriani, P. Magillo, E. Puppo, “Efficient implementation of multi-triangulations”, Proceedings of IEEE Visualization ’97, pp. 103 – 110, 1997.

[ElSa99] J. El-Sana, A. Varshney, “Generalized View-Dependent Simplification”, Proc. of EUROGRAPHICS’99, pp. 131-137, 1999.

[ElSa99b]] J. El-Sana, E. Azanli, A. Varshney, “Skip Strips: Maintaining triangle strips for view-dependent rendering”, Proc. of Visualization’99, pp. 131-137, 1999.

[Heck97] P. Heckbert, M. Garland, “Survey of Polygonal Surface Simplification Algorithms”, Siggraph’97 Course Notes, 1997.

[Hopp96] H. Hoppe, “Progressive meshes”, Proceedings of SIGGRAPH’96, pp. 99-108, 1996.

[Hopp97] H. Hoppe, "View-dependent refinement of progressive meshes", Proc. of SIGGRAPH'97, pp. 189-198, 1997.

[Hopp98] H. Hoppe, “Efficient implementation of progressive meshes”, Computer & Graphics, 22(1), pp. 27-36, 1998.

[Lint99] B. Lintermann, O. Deussen. “Interactive modeling of plants”. IEEE Computer Graphics and Applications, 19(1), 1999.

[Lueb97] D. Luebke and C. Erikson, “View-Dependent Simplification of Arbitrary Polygonal Environments”, Proc of SIGGRAPH’97, pp. 202-210, 1997.

[Mars97] D. Marshall, D. Fussell, A. T. Campbell III, "Multiresolution Rendering of Complex Botanical Scenes", Graphics Interface '97, pp. 97-104, 1997.

[Max96] N. Max, K. Ohsaki. “Rendering trees from precomputed Z-buffer views”, Eurographics Workshop on Rendering 1996, pp. 165-174, 1996.

[Popo97] J. Popovic, H.Hoppe, “Progressive simplicial complexes”, Proceedings of SIGGRAPH’97, pp. 217-224, 1997.

[Prus90] P. Prusinkiewicz, A. Lindenmayer, “The algorithmic beauty of plants”, New York, Ed. Springer-Verlag, 1990.

[Pupp97] E. Puppo, R. Scopigno, “Simplification, LOD and Multiresolution – Principles and Applications”, Eurographics’97, Tutorial Notes, 1997.

[Pupp98] E. Puppo, “Variable resolution triangulations” Computational Geometry, 11(3-4), pp. 219-238., 1998.

[Remo02] I. Remolar, M. Chover, O. Belmonte, J. Ribelles, C. Rebollo, “Geometric Simplification of Foliage”, Eurographics'02 Short Presentations, pp. 397-404, 2002.

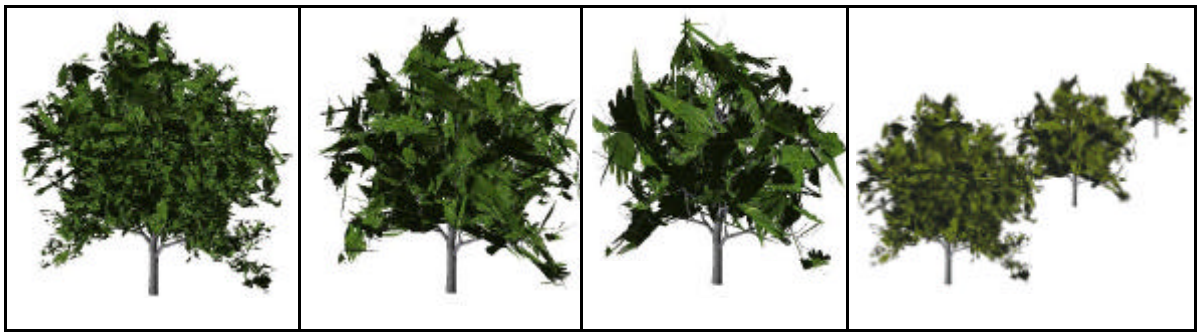
[Ribe02] J. Ribelles, A. López, Ó. Belmonte, I. Remolar, M. Chover. "Multiresolution Modelling of Arbitrary Polygonal Surfaces: A Characterization", Computers & Graphics, 26(3),pp.449-462, 2002.

[Scha98] G. Schaufler. “Per-object image warping with layered impostors.” Eurographics Rendering Workshop 1998, pp. 145-156, 1998.

[Shad96] J. Shade, D. Lischinski, D. Salesin, T. DeRose, J. Snyder, “Hierarchical image Caching for Accelerated Walkthroughs of Complex Environments”, Proceedings of Siggraph’96, pp. 75-82, 1996.

[Xia96] J. Xia., A. Varshney, “Dynamic view-dependent simplification for polygonal models”, Proc. of IEEE Visualization’96, pp. 327-334, 1996.

[Xia97] J. Xia, J. EL-Sana, A. Varshney, “Adaptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models”, IEEE Transactions on Visualizations and Computer Graphics 3(2), pp. 171-183, 1997.



a) b) c) d)

Figure 10: Different uniform levels of detail of the same tree. a) 13.420 polygons, b) 1.558 polygons and c) 472 polygons. These levels of detail are shown in d) depending on the distance to the viewer.

a) b) c) d)

Figure 11: View dependent levels of detail. Criteria used for determining the interest area is a sphere. a) and b) 27.136 polygons, c) and d) 13.680 polygons.

a) b) c) d)

Figure 12: View dependent levels of detail. Criteria used: a plane. a) and b) 24 768 polygons, c) and d)18.406 polygons.

Figure 13: Approximation movement towards a tree