EUROGRAPHICS 2002/ I. Navazo and Ph. Slusallek
(Guest Editors)

Geometric Simplification of Foliage

I. Remolar, M. Chover, Ó. Belmonte, J. Ribelles, C. Rebollo

Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I, Castellón, Spain

Abstract

One of the most important challenges in real-time rendering of outdoor scenes is the representation of vegetation. This is due to the vast amount of polygons that are used to model vegetable species. The present automatic simplification algorithms produce acceptable results in the trunks and the branches, but not in the foliage of a tree. After simplification with the existing methods, the trees seem to be less leafy. In this paper, a new automatic simplification algorithm for this part of the tree is presented, the Foliage Simplification Algorithm. It diminishes the number of polygons in the crown, while maintaining the appearance. It uses a new method, leaf collapse: two leaves disappear to create a new one. The leaves obtained after collapsing preserve an area similar to that of the collapsed leaves

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 modelling has been widely investigated [1,2], and its representation is very realistic (Figure 1). However, tree models are formed by such a vast number of polygons that real-time visualisation of scenes with trees is practically impossible.

Real-time visualisation 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 [3,4,5]. In some applications, however, geometry is necessary. 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. One of the existing methods to obtain this objective is the application of automatic simplification algorithms [6,7].

Many automatic simplification methods have appeared up to now. Applying them to trees, these obtain acceptable results with the meshes of polygons that represent the trunk and the branches. However, they do not work properly with the foliage. The existing simplification methods generally eliminate polygons, so that the appearance of the crown after an automatic process of simplification is that it has been pruned. The number of leaves is less than before, so the tree appears less leafy. The images obtained with these methods are not very realistic and for this reason it is necessary to introduce new solutions.

Figure 1: *Aesculus hippocastanum. Tree modelled with the commercial modelling tool Xfrog. 192.179 triangles*

The method presented in this paper for the automatic simplification of foliage diminishes the number of polygons that form the crown, while maintaining its leafy appearance. 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 the way of determining the pair of leaves that will be simplified to create a new one.

2. Related Work

Because previous work on geometric simplification has recently been reviewed in several papers [6,7], in this article we review the different existing methods of simplification. We analyse the results that would be produced on the mesh of isolated triangles that constitutes the foliage of the trees.

According to [7], one of the methods traditionally used to generate simplified versions of an object, is the *manual method*. The user generates several levels of detail by hand. Simplified versions of trees and plants can be obtained, in the case of using L-systems, by limiting the number of polygons at the time of generating the object. This is one of the most widely used methods for tree geometry simplification. The commercial software Xfrog [2] has an additional tool, denominated XfrogMLOD, to generate these levels of detail. The user determines the number of leaves or branches that conform the tree during its modelling. Varying this parameter, different levels of detail of a same tree are obtained. But these tools cannot simplify any geometric description of a tree. They only simplify trees that have been generated with that software.

Another method is *Vertex Clustering* [8,9]. It partitions the vertex set spatially into clusters and unifies all vertices within the same cluster. This produces simplified trees that appear to have been pruned.

Region Merging [10] and *Wavelet Decomposition* [11] methods do not work properly with meshes of isolated polygons. Region Merging is generally restricted to manifold surfaces. Wavelet Decomposition is adequate for surfaces with subdivision connectivity.

Vertex Decimation [12,13] does not produce good results either. In each step of the decimation process, a vertex is selected for removal, all the faces adjacent to that vertex are removed from the model, and the resulting hole is re-triangulated. In our case, each leaf is formed by two triangles with an image textured on it. If one of them is eliminated, it would cause the image to be disfigured. The *Iterative Contraction* [14,15,16] method would produce the same effects as those mentioned with regard to vertex decimation.

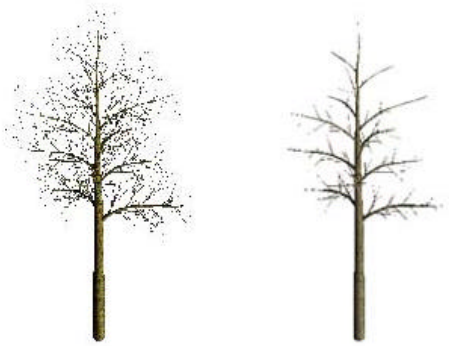
The *Foliage Simplification Algorithm* has been developed in order to generate different levels of detail of a same tree without losing similarity with the original model.

3. Tree geometry simplification

The trees used in our study are modelled by the Xfrog application [2] (Figure 1). They are very realistic, but are generally formed by more than 50.000 polygons each. This is a disadvantage when it comes to generating images in an interactive way.

The trees can be separated into two different parts: the solid component of the tree, i.e. the trunk and the branches, and the sparse component, the foliage or leaves. The trunk and the branches are represented by triangle meshes and the foliage by a set of isolated polygons where each of the leaves is a textured quadrilateral.

The trunk is formed by a set of meshes of polygons. A great number of automatic simplification algorithms existing in the literature deal with this type of objects. In this work, the algorithm proposed by Garland and Heckbert [14], *qslim* has been used. This algorithm is public domain and it is available at http://graphics.cs.uiuc.edu/~garland/research/quadrics.html (Figure 2).



a) b)

Figure 2: *a) Image of the original trunk formed by 47.691 polygons and b) image of a simplified trunk with 1.999 polygons.*

Secondarily, the foliage of the tree is formed by a set of independent polygons. The automatic simplification algorithms that have appeared up to now do not work properly with this type of representation. Figure 3 shows the crown of a tree

after application of the automatic algorithm *qslim*. It can be seen that, in this image, the tree is less leafy.

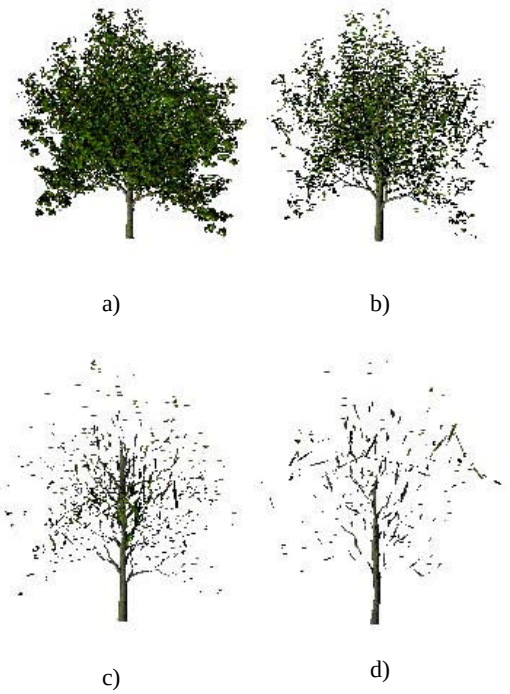


Figure 3: *Example of a simplified foliage from a) 20.376 leaves to b) 6.710, c) 779 and d) 236 leaves.*

Foliage Simplification Algorithm, *FSA*, has been defined with the purpose of diminishing the number of polygons that form the foliage of the tree without losing the leafy appearance. The algorithm is described below.

4. Foliage Simplification Algorithm

The tree leaves defined with the Xfrog application are represented by quadrilaterals formed by two triangles. The final aspect is obtained by texturing these quadrilaterals with the image of a leaf. The method of simplification presented in this paper repeatedly selects a pair of leaves, which minimises an error function. These leaves disappear and a new one is obtained. The collapsed leaves are eliminated from the list of candidates, and next, the new leaf is evaluated with the leaves that remain in the foliage.

The main idea of this simplification algorithm is that the leaf obtained after collapsing maintains an area similar to that of the collapsed leaves. This is done in order to preserve the appearance of the foliage when the number of leaves is reduced.

The simplification method is characterised 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. These two questions are discussed in the following sections

4.1 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.

Assuming that l_1 and l_2 are two leaves pertaining to a certain level of detail, the error function is as follows:

$$e(l_1,l_2) = k_1 \times d_H(l_1,l_2) + k_2 \times c(l_1,l_2)$$

where $d_H(l_1, l_2)$ is the distance between leaves, and $c(l_1, l_2)$ the planarity level. Measurement of the distance between leaves is performed according to *Hausdorff*. This measurement makes geometric comparisons between two sets of points, using the shortest distance between a point x and a set of points Y:

$$d(x,Y) = min_{y \in Y} d(x,y)$$

where $d(x, y)$ is a measurement between a pair of points. The distance of *Hausdorff* is defined as:

$$d(X,Y) = \frac{\sum_{x \in X} d(x,Y) + \sum_{y \in Y} d(y,X)}{|X| + |Y|}$$

In addition, we also measured the planarity between two leaves. This was done by comparing the angles formed by the normal vectors of the leaves. Two constants were used, k1 and k2 in the function. In this way, the relevance of the two criteria can be changed by varying these constants.

The planarity criterion would give back values within an interval of 0 and 1, because it is based on the scalar product of the normal of the leaves. Similarly, the criterion of the distance is normalised with respect to the diameter of the sphere surrounding the crown. All this means that the error function moves in the range [0,1].

4.2 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. For this reason, the two vertices of each leaf that are furthest from the other leaf would be chosen. This method will allow us to maintain an area similar to the two original leaves. However, the two triangles that will form the new leaf are not generally in the same plane.

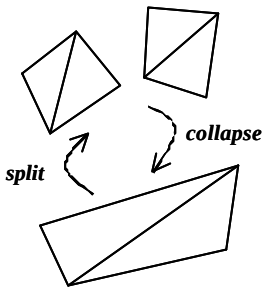


Figure 4: *Simplification of two leaves to create a new one. The vertices of the original leaves remain.*

4.3 Algorithm Overview

The main data structure of the algorithm is the *Leaf* class, shown in Figure 5. Each of the leaves in the foliage is represented by an object of this class.

```
Class Leaf
{       int id;
        int Vertex[4];
        float Normal[3];
        float error;
        int id_couple;
        int lnumber;
        bool exists;   }
```

Figure 5: *Principal C++ data structure.*

Firstly, all the polygons that make up the foliage are possible candidates to be simplified. For this reason a flag, *exists*, has been introduced which initially would be *true* in all the polygons.

Each of the leaves will be evaluated with the rest according to the error function $\varepsilon(l_1, l_2)$. Two data will be stored in them:

- the leaf that makes the error function as low as possible, *id_couple*, and
- the value of this function, *error*.

The lnumber field indicates the number of leaves that have been collapsed to create this leaf. In order to prevent some leaves from growing in an unbalanced form with respect to others, we imposed the condition that two leaves will only be collapsed if the values of their *lnumber* differ by one. In this way, the coexistence of excessively large leaves and the original, much smaller, leaves is avoided.

When all the leaves have been evaluated, the pair of leaves that make the lowest error function is chosen. This selected pair will be taken away from the candidate leaves putting flag *exists* to *f*alse, and the new leaf will enter on this list.

Considering L_s as the set of leaves that initially form the crown of the tree, the core of the traversal algorithm is summarised in Figure 6.

```
for each leaf l1 ∈ Ls {

l1.error = MAXERROR; // inicialization

 for each leaf l2 ∈ Ls {
 if (l2.exists == true) && (l2 != l1){
  if (abs(l2.lnumber – l1.lnumber))< 2
   {

   P = Planarity (l1,l2);
   D = Haussdorf (l1,l2);
   ε = Criterium (P,D);

   If (l1.error > ε){
      l1.error = ε;
      l1.id_couple = l2; }
   }
 }
}
```

Figure 6: *Pseudocode of the algorithm that calculates the error*

When a leaf collapse is performed, the new leaf will be evaluated with the rest of the leaves in the same way as described above. The same process will be done for the candidate leaves that store in the *id_couple* field one of the two leaves that have just been collapsed.

The number of leaves that make up the simplified crown is determined a priori by the user. This will condition the number of iterations of the algorithm.

5. Results

The method developed was implemented with OpenGL on a PC with Windows 2000 operating system. The computer used was a dual Pentium III Xeon at 1.5GHz. with an NVIDIA Quadro2 Pro graphics processor with 64MB.

The distance between leaves has been considered a more important criterion than planarity in our experiments. In this way, our results have been obtained with $k_1 = 0,8$ and $k_2 = 0,2$.

Figure 7 shows the three tree models used for the experiments. They have been simplified with the FSA algorithm, and the results are presented in Figure 8. It can be observed that the images obtained maintain the appearance although their leaf number diminishes. In these images, the trunk has been simplified from 38.781 polygons to 12.771 polygons in Figures 8a.1, 8b.1 and 8c.1, 1.482 polygons in 8a.2, 8b.2 and 8c.2, and 449 at the rest.

Figure 9 shows some VRML world scenes where appears the tree shown in Figure 7a. Four different levels of detail have been used to design this VRML world, that is available at http://graficos.uji.es/trees/.

6. Future Work

In this work, we have studied the geometric simplification of foliage, accepting the results of the simplification algorithms on the structure of the trunk and the branches. But these algorithms prune the branches when simplifying this structure. The next step is to develop a simplification algorithm that does not cut the branches and limbs. The main idea is that the algorithm determines its skeleton and eliminates polygons while still maintaining it.

Another line of research we are currently working on is the development of a continuous multiresolution model. Some authors [6,7,17] have classified multiresolution models into basically two groups: discrete and continuous. Discrete models contain a finite number of levels of detail and a control mechanism to determine which is the most adequate in each moment. It is possible to obtain different levels of detail of the same tree with our algorithm. This would constitute a discrete multiresolution model. On the other hand, the continuous models capture a vast range of approximations of an object, virtually continuously. They offer the possibility of adapting the level of detail in real time. This accelerates the visualisation. Some of this type of multiresolution models can represent an object with variable resolution [18], that is, different resolutions can coexist in different regions of the rendered object.

Hoppe presents a multiresolution model with these characteristics in [18], based on the simplification operation *edge collapse*. Thus, we are developing a continuous multiresolution model based on the *leaf collapse* operation. This is a continuous model that allows us to represent the foliage with variable resolution. Our multiresolution model of the trees will permit us to adjust the level of detail to the application requirements. In this way, fewer polygons will be used to draw distant objects. Furthermore, the model of the crown will allow us to represent the foliage at variable resolution. For example, when trees are close to the viewer, the front parts will be represented with more detail, and the back parts with less detail.

For more information and colour images, please visit http://graficos.uji.es/trees/.

Acknowledgements

This work was supported by the Spanish Ministry of Science and Technology grants TIC1999-0510-C02-02 and TIC2001-2416-C03-02

References

1. P. Prusinkiewicz, A. Lindenmayer, “The algorithmic beauty of plants”, New York, Ed. Springer-Verlag, 1990.
2. B. Lintermann, O. Deussen. “Interactive modelling of plants”, IEEE Computer Graphics and Applications, 19(1), 1999.
3. N. Max, K. Ohsaki. “Rendering trees from precomputed Z-buffer views”, Eurographics Workshop on Rendering 1996, pp. 165-174, 1996.
4. D. Marshall, D. Fussell, A. T. Campbell III, "Multiresolution Rendering of Complex Botanical Scenes", Graphics Interface '97, pp. 97--104, 1997.
5. A. Jakulin. “Interactive Vegetation Rendering with Slicing and Blending”, Eurographics’2000, Short presentations, 2000.
6. E. Puppo, R. Scopigno, “Simplification, LOD and Multiresolution – Principles and Applications”, Eurographics’97, Tutorial Notes, 1997.
7. P. Heckbert, M. Garland, “Survey of Polygonal Surface Simplification Algorithms”, Siggraph’97 Course Notes, 1997.
8. J. Rossignac, P. Borrell. “Multi-resolution 3D approximations form rendering complex scenes”, Modelling in Computer Graphics: Methods and Applications, pp. 455-465, 1993.
9. K. Low, T.-S. Tan, “Model simplification using vertex-clustering”, 1997 Symposium on Interactive 3D Graphics, ACM Siggraph, 1997.

10. A. D. Kalvin, R. H. Taylor. “Superfaces: Polygonal mesh simplification with bounded error”, IEEE Computer Graphics and Appl., 16(3), May 1996.

11. E.J. Stollnitz, T. D. DeRose, D. H. Salesin, “Wavelets for Computer Graphics: Theory and Applications”, Morgann Kaufmann, San Francisco, CA, 1996.

12. W.J. Schroeder, “A topology modifying progressive decimation algorithm”, IEEE Visualization 97 Conference Proceedings, pp. 205-212, 545, 1997.

13. A. Ciamplaini, P. Cignoni, C. Montani, R. Scopigno, “Multiresolution decimation based on global error”, The Visual Computer, 13(5): pp. 228-246, 1997.

14. M. Garland, P.S. Heckbert, “Surface simplification using quadric error metrics”; Siggraph’97 Proc., pp. 209-216, 1998.

15. P. Lindstrom, G. Turk, “Fast and memory efficient polygonal simplification”, IEEE Visualization 98 Conference Proceedings, pp. 279-286, 544, 1998.

16. H. Hoppe, “Progressive meshes”, Siggraph’96 Proc., pp. 99-108, 1996.

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

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

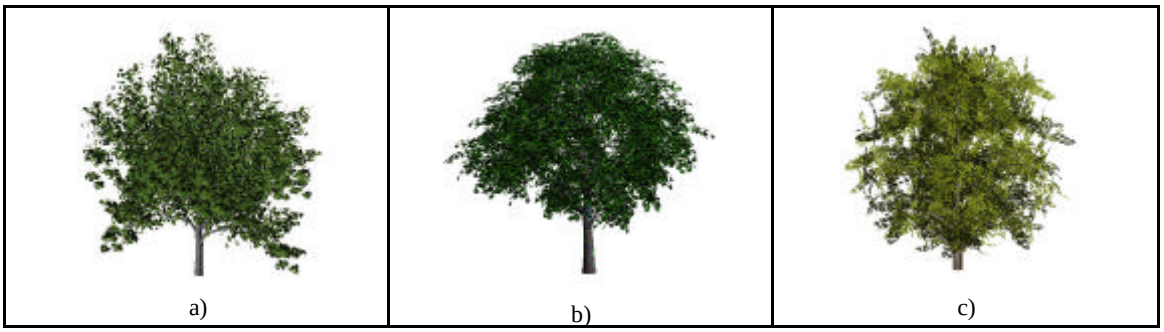


Figure 7: *Tree models used for the test. a) English Oak: 20.376 leaves b) Aesculus hippocastanum: 29.535 leaves, c) Sorbus aucuparia: 24.840 leaves.*

Figure 8: *Results of simplification of the trees shown in: Figure 7.a: a.1) 6.710 , a.2) 779 and a.3) 236 leaves. Figure 7.b: b.1) 15.612, b.2) 8.783 and b.3) 4.391 leaves. Figure 7.c: c.1) 6.996, c.2) 1.026 and c.3) 343 leaves.*

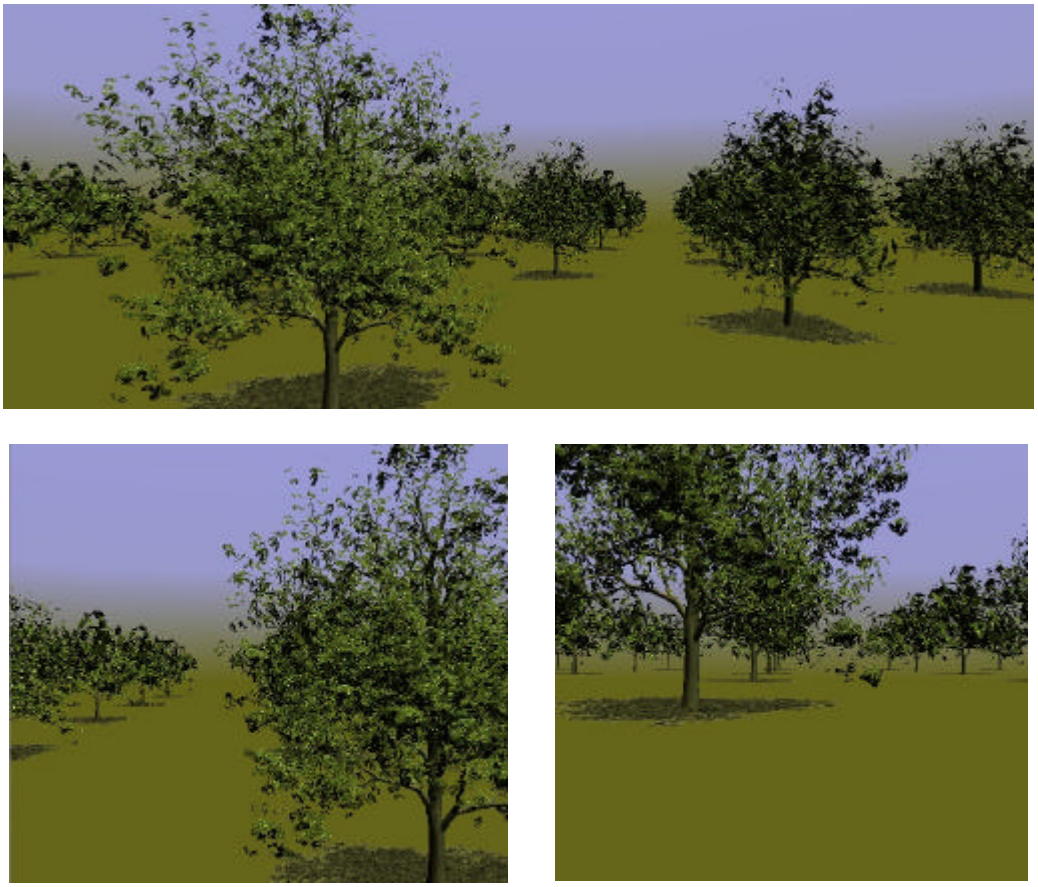


Figure 9: *Some VRML world scenes using a LOD node with four different levels of detail.*