Modelling tree structures using a single polygonal mesh*

J. Lluch, R. Vivó, C. Monserrat

Computer Graphics Section

Department of Information Systems and Computation

Valencia University of Technology

Camino de Vera s/n

46022 Valencia (Spain)

e-mail: jlluch@dsic.upv.es

* This work was supported by TIC2002-04166-C03-01

Abstract. This article presents a method to obtain a polygonal mesh representing the ramified structure of a tree. We use a method based on L-systems to model the tree. A refining process is applied once the model of the tree is obtained. Soft transitions are achieved in the areas of the tree where there are divisions. In this way, we avoid the possible superimposition and discontinuity problems in geometry. As a result of this process, a continuous polygonal mesh is obtained. This mesh represents all the branches of the tree, allowing the homogeneous application of textures and algorithms to simplify meshes.

Keywords: Computer graphics, polygonal mesh, modelling of natural phenomena, modelling of vegetable species.

1 Introduction

The synthetic creation of plants represents a great challenge for people working in computer graphics. Plants that are seen at a close distance have to display soft transitions in the branches and a high level of detail at any scale. As a result of this, all plant models are bound to be characterised by being very complicated and having a non-continuous representation.

In this article a method which permits modelling of a tree or plant, or any kind of branched structure, is presented. It uses a single polygonal mesh that can be easily simplified. However, the application of this kind of technique presents difficulties storing different levels of detail and the transition among them. This is why a soft transition multiresolution model suitable for branched structures should be developed.

In the following section, the main current methods for tree modelling are analysed, together with the kind of geometry used to describe them. In addition, the visualization acceleration techniques used by different authors are studied. The third section explains the algorithm of mesh production, which is divided into three parts: tree representation, node refining, and obtaining the polygonal model. In the fourth section, we present our results. Finally, in the last section we present our conclusions and future enhancements of our method.

2 Background

Modelling vegetal elements is a field inside computer graphics, which has been the object of study by many authors. Among the different approaches, we highlight two: modelling based on botanical principles; and modelling that searches for realism without necessarily copying natural phenomena. It may also be observed from the study of different models that they are either defined through formal methods to describe the branches of the structure; or through techniques obtained from the observation of different elements.

Initially, the main motive for modelling vegetal elements was biological. The first steps in this field were taken by the biologist Aristid Lindenmayer, who developed the L-system [1] for its application in cell interaction. The first application of L-system to tree modelling in computer graphics can be found in [2]. This simple model was later developed by Prusinkiewicz [3][4] for its application to vegetal elements. The model is based on the concepts of database amplification and rewriting rules. Starting from an initial

module called axiom, and applying the rules in a parallel and consecutive way, a chain of symbols is obtained which can be interpreted graphically. The symbols making up the language can have some associated parameters. The model incorporates context-sensitive L-systems, which permits the simulation of endogenous phenomena in individuals, and the possibility of applying the rules randomly to create different individuals of the same species. This model has been one of the most commonly used and it has been extended for the simulation of different phenomena, such as pruning (topiary and obstacles) [5], or the interactivity of plant development with its environment (competition for light, space, or resources) [6].

Additionally, particle systems have also been used to model trees and improve their rendering. The method by Reeves [7] models a tree by starting from the trunk and creating the branches recursively. The process output is a tree data structure where each node defines one segment of a branch.

Bloomenthal [8] introduces a method that represents trees using points and connections. This method uses generalized cylinders to represent the branches. It supports connecting branching limbs with a free-form surface. Although this method solves branch connection in a nice way, it is a particular and reduced model because it can only pass information by lineage. The resulting tree is not a homogeneous polygonal mesh.

Oppenheimer [9] uses fractals for tree modelling. There is a parameterisation to regulate the relation between the nodes, the angle between the trunk and the branches, and the size of the branches and the trunk, how the branches are narrowed, or the number of ramifications per branch. These parameters may have random variations to avoid fractal auto-similarity.

De Reffye et al. [10] present a model entirely based on botanical principles taking into account the following information: the way the trees grow, how they occupy the space, or where and how the leaves, flowers and fruits are located. This method models the bud activity, which at any given moment may: blossom, stop its growth, become an internode, or die. These events happen according to stochastic laws that characterise each variety or each species. The procedure output is a set of elements (internodes, flowers, leaves, and fruits) which make up the structure of the tree.

Weber and Penn [11] put forward a method based on the parameterisation of certain features of trees obtained after observing different species. The model is similar to that of Oppenheimer, although it avoids auto-similarity so as not to limit the representation potential of the method. Children branches inherit some features from their ancestors but others may be completely different.

Lintermann and Deussen [12] propose a mixture of geometrical modelling techniques and rule applications. The tree is represented by a graph. Each graph component includes data and algorithms for development. There are three kinds of components: graphic models, multiplication of other components, and global modelling techniques.

The model used in this paper to create trees is based on parametric L-systems [13]. The main differences lie in the use of random variables in the system and the creation of special modules to permit the control of chain derivation.

2.1 Resulting Geometry

There is a wide range of models to generate trees. However none creates a homogeneous geometrical structure to describe the generated objects. The results of the studied methods are combinations of different geometrical models. Every model represents a branch or a branch segment. The most commonly used primitives are: truncated cones, generalised cylinders or polygon meshes for each branch.

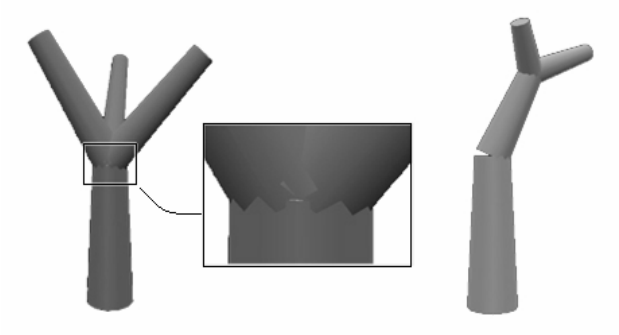


Figure 1*. Visibility and continuity problems in junctions where primitives meet*

The use of some methods (truncated cones, cylinders, mesh per each branch) to obtain curved branches may provoke some discontinuities. Problems may also arise when representing junctions, since some polygons of different branches may superimpose, with the consequent visibility problems (see Figure 1). For this reason, we propose the generation of a single mesh to represent the structure of a tree.

2.2 Acceleration of Visualization

The large amount of geometry necessary to model natural environments causes a number of problems at rendering time. Not even the most powerful graphic accelerators are able to generate high enough frame

rates for the interactive rendering of a scene showing a few trees, if a brute-force algorithm is used. That is why it is necessary to use techniques to accelerate the visualisation of this kind of scene.

Nowadays, one of the most important applications of tree modelling is the rendering of ecosystems. To achieve this, it is necessary to represent simultaneously a large number of trees. In this kind of application, the rapid visualisation of each model is very important. There are many different methods for the rendering acceleration:

- Weber and Penn [11] put forward the reinterpretation of geometry depending on the distance to the observer, drawing branches as lines — or not drawing them — as the distance increases.
- House et al. [14] use techniques of level of detail as an acceleration method.
- Max [15] uses textures to replace geometry.

Obtaining of a single mesh will allow the use of a multiresolution structure. It can be adapted to the branch structure of the model.

3 Materials and Methods

The construction of a tree made from a single triangle mesh by interpreting a chain resulting from the derivation of an L-system undergoes three different phases:

- The phase of modeling the tree.
- The phase of obtaining the polygonal model.
- The phase of refining the tree nodes.

The result of this last phase is a single 3D triangular mesh describing the tree built with the RL-system.

3.1 Representation of the Tree

The interpretation of the chain is based on the ‘turtle’ metaphor normally used when developing graphic applications in the programming language LOGO [4]. A position, an orientation, and a stack to store previous positions and orientations characterise this turtle.

The turtle orientation can be varied by means of rotation movements on each of the 3D orthogonal axes associated with the turtle, whereas the movement is always generated in the direction marked by the H

axis (heading). In each displacement, the turtle creates a contour (either elliptical or circular) in the initial position and another in the final position. The size of that contour will depend on the thickness and inclination of the current branch. Contours are later used to produce the geometry of the tree.

To store the contours generated by the turtle, a hierarchical tree structure is created. Each node of the tree consists of: the contour generated by the turtle, a transformation matrix, a pointer to the next sibling node, a pointer to the first child node and a pointer to the parent node (see Figure 2). The matrix contains the rotation and translation operations to enable the turtle to go from the previous node (parent node) to the current one. The parent node is the one from which the turtle started (start movement point) before reaching the current node (final movement point). All the children nodes from the same parent are linked through a list which is joined according to the order in which they were generated. A parent node has direct access only to the first child node.

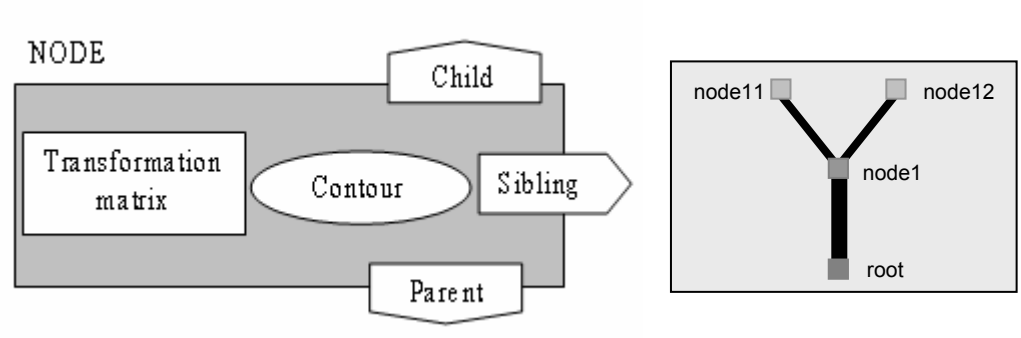


***Figure 2**. Content of a node*

Starting from this hierarchical structure that has been obtained from the interpretation of the chain of symbols produced by the L-system, a triangular mesh is generated which respects the contour shapes and the topology established by the structure obtained in this phase. The mesh is generated by joining the vertices of the two contours associated to each branch.

3.2 Obtaining the Polygonal Model

We have developed a software library for the construction of the polygonal model. It enables the creation of a triangular mesh from a sequence of contours. Attempts have been made since 1970s to reconstruct volumes from a group of flat contours. The importance of this algorithm is a consequence of applications in both engineering (Computer Aided Design) and medicine. It is in the latter where development has

been fostered with the appearance in the early seventies of examination instruments such as nuclear magnetic resonance (NMR) or computerised axial tomography (CAT).

The library developed in [13] solves the problems identified by Meyers and Skinner [16], and improves the efficiency of other reconstruction methods. The reconstruction library is applied when generating triangle meshes between pairs of consecutive sections. In the case of the algorithm presented in this paper, the lower section is made up of one contour from the father node. All the children nodes of the current father node make up the upper section. The section construction and the mesh result are shown on Figure 3.

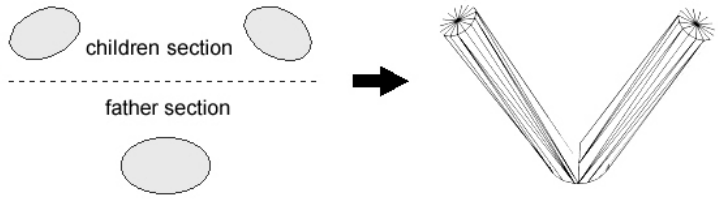


Figure 3. *Generation of the triangular mesh between two adjacent sections*

As Figure 3 shows, the direct application of the reconstruction algorithm for generating a triangular mesh from the structure generated in the phase of the representation of the tree shows a geometry that is visually erroneous at the meeting point with the father node. To avoid this problem, we propose a refining algorithm.

3.3 Refining the Junctions

An algorithm called "refining by intervals" has been developed to correct the geometrical errors produced at the junctions of the branches (tree nodes). This method enables the addition of a group of new contours, equi-spaced in height, to the structure between the parent node and its children. These contours are added from the meeting point of the ramifications until they are completely separated (see Figure 4). The height of the first contour is computed by adding a constant amount to the height of the parent contour. The heights of subsequent contours are computed adding the same constant increment. The conturs thus generated are equally spaced in height.

To achieve the refining by intervals, the tree branches are asssumed to be represented through contours defining cylinders or truncated cones with equal height and width to the length and thickness of the real branch. Following this assumption, the generation of intermediate contours follows a three-step sequence:

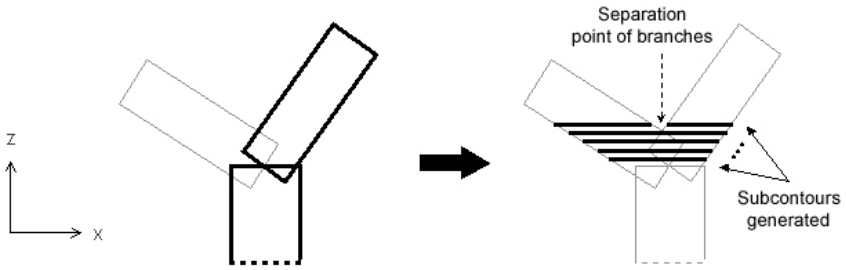


Figure 4. *Contour insertion made by the method of refining at intervals. At each height the cylinders are intersected by a horizontal cutting plane that moves upwards at equi-spaced intervals.*

- Calculation of the parameters of the ellipses obtained by sectioning the cylinders at the specified height (minor radius r_e, major radius R_e, centre C and rotation δ).

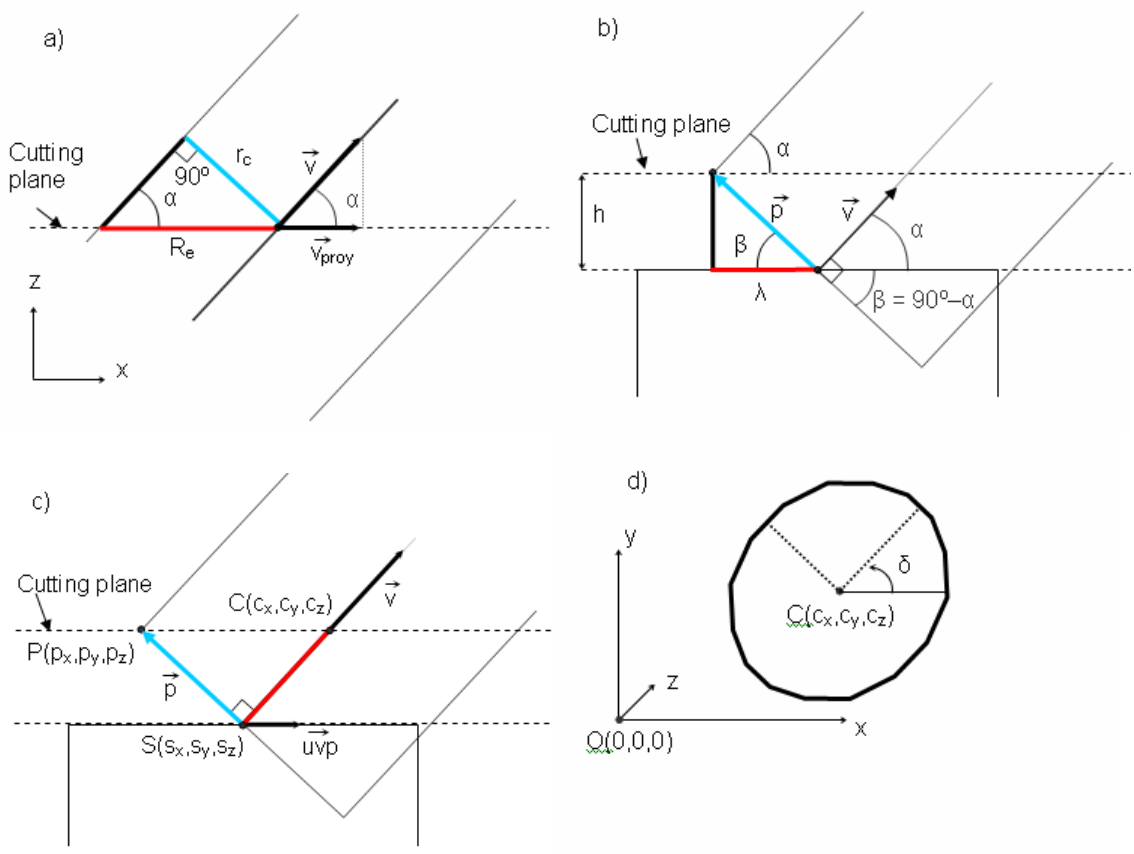


Figure 5. *Front orthographic views of the cylinders (a,b,c) and up view of the ellipse obtained (d). This figure shows the data needed to obtain the parameters of the ellipse.*

First, the ellipse radii are calculated (1) (see Figure 5, pictures a and b), where $\vec{v}$ is the vector along the axis of one of the branch cylinders, $\vec{p}$ is the vector perpendicular to $\vec{v}$ from the center of the cylinder to the cylinder's edge, $\overrightarrow{uvp}$ is the unit vector on the direction of the projection of $\vec{v}$ onto the base of the other cylinder, λ is the size of this projection and α is the angle between $\vec{v}$ and $\overrightarrow{uvp}$:

$$r_e = r_c \,,\; R_e = \frac{r_e}{\sin(\alpha)} \tag{1}$$

To obtain $\vec{p}$ coordinates, we know $\vec{p} \perp \vec{v}$ (see Figure 5, pictures b and c):

$$p_x \cdot v_x + p_y \cdot v_y + p_z \cdot v_z = 0 \,, \cos(\beta) = \frac{\lambda}{r_e}, \tag{2}$$

$$p_x = -\lambda \cdot \overrightarrow{uvp}_x,$$

$$p_y = -\lambda \cdot \overrightarrow{uvp}_y.$$

Then, the ellipse centre is calculated, from the height *h* (Figure 5b) where it must be moved to, and the angle of rotation necessary to rotate it to its final position (Figure 5d), for simplicity we suppose that S is at the origin:

$$h = p_z = \frac{-\left(p_x \cdot v_x + p_y \cdot v_y\right)}{v_z}, \; \delta = \arccos\left(\frac{v_x}{\sqrt{v_x^2 + v_y^2 + v_z^2}}\right). \tag{3}$$

- Generation of a single contour, from the ellipses obtained, for each of the interval sets from the parent node to the point at which all the ramifications separate (see Figure 6):

  First, the ellipses are approximated by polygons and the vertices of all ellipses are sorted in counterclockwise order. The number of vertices is selected depending on the size of the branch and the visual accuracy needed at the branches connections

  Starting at a point belonging to an ellipse and external to others we follow the edges of the current ellipse until we find an intersection with another ellipse

  Then, we create a new edge from the first vertex of the current edge to the last vertex of the edge intersected of the other ellipse. Then we follow the edges of the new ellipse (see Figure 6 a)

If the new edge created intersects with other edge, we have to create another edge until it does not intersect with other edge. In figure 6b the new edge (dotted line) intersects with an edge of ellipse 3, and then we create a valid edge from ellipse 1 to ellipse 3.

When this process is made for all contours, we obtain a structure like the one shown in figure 6c.

- Finally, a node is constructed with every contour generated in the previous step. This is added to the tree structure generated by the interpretation of the entry chain to the current algorithm. The addition of this node is made with respect to its structure (see figure 7).

The number of inserted nodes will depend exclusively on the inclination of the cylinders. If their inclination is small, with respect to the plane defined by the contour of the parent node, the lack of intersections between the ellipses will be detected by the method in a few iterations. Subsequently, the number of nodes to insert will be smaller. On the contrary, if the cylinders show a large inclination angle, the separation point will be quite far from their base. Therefore, a lot of intermediate nodes will have to be inserted for the same interval value.

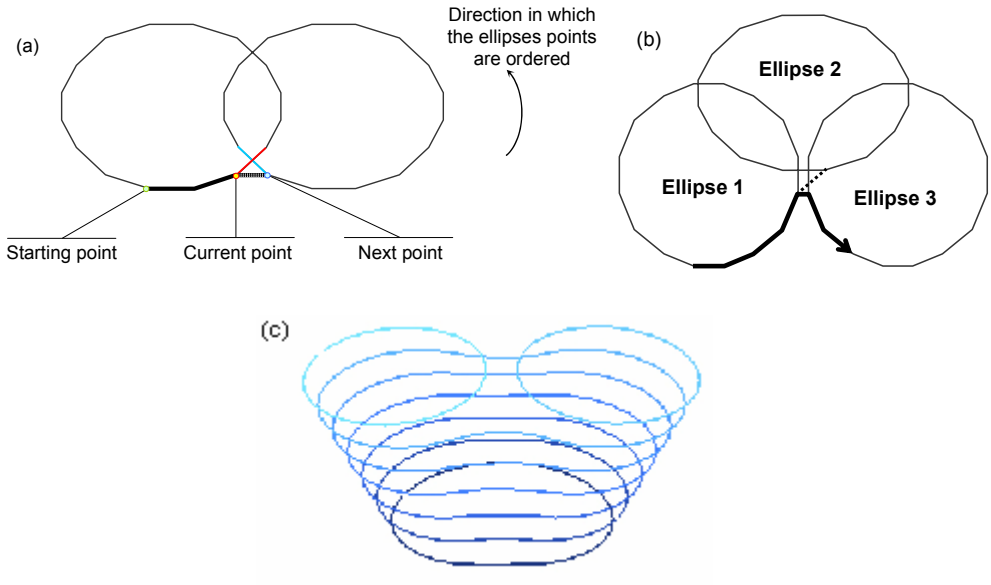


Figure 6. *Generation of a single contour from the detection of intersections between edges*

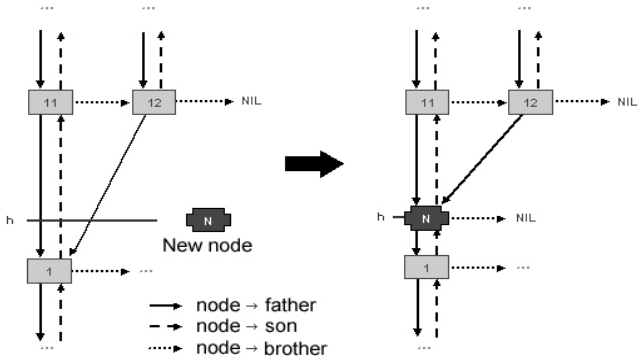


Figure 7. *Insertion of the contour joined to the tree structure.*

4 Results

The result of the algorithm described in this article is a single triangle mesh representing in three dimensions the tree described by the chain of symbols taken as input. This mesh avoids visibility and continuity problems as shown in Figure 8 compared to Figure 1. The result of applying the reconstruction algorithm developed is shown in Figure 9. These results are shown both with, and without, application of the refining method. The result of the reconstruction with refining is visually better than the one without. Table 1 shows a comparison of the number of triangles generated with the method of refining at intervals compared to the direct method. This table shows that the visual improvement obtained by the method of refining at intervals is made by increasing the spatial cost by three times. Contour resolution refers to the number of vertices generated for each contour.

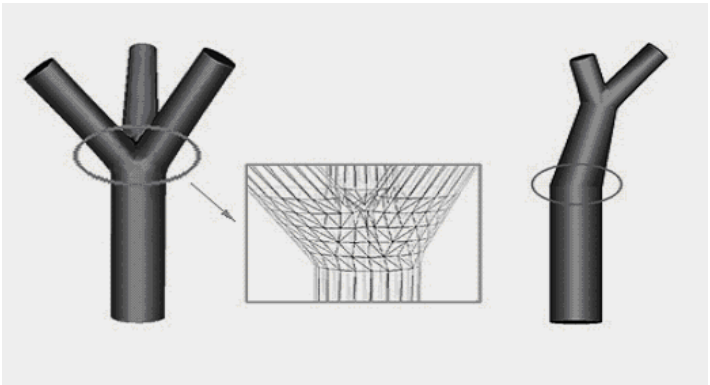


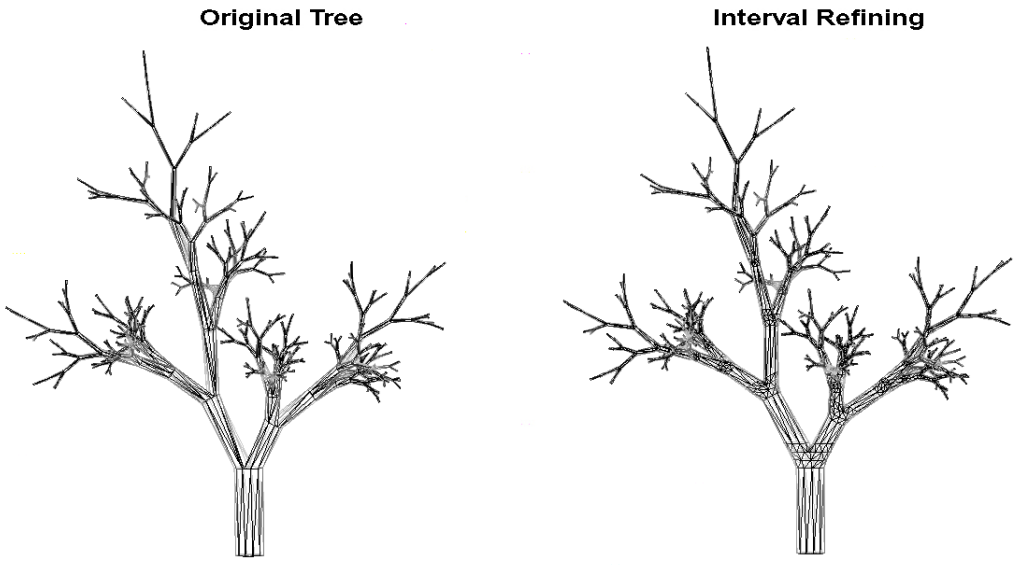
Figure 8*. The single mesh gives solutions to the visibility and continuity problems from previous pictures.*

This increase in the spatial cost suggests the use of a simplification algorithm to reduce the amount of geometry generated for the above meshes. For the reconstructions obtained with the algorithm described

in this article, a decimation algorithm [17] can be applied that achieves up to an 80% reduction in the number of triangles representing the mesh without affecting the visual improvement obtained with the method of refining at intervals. This process is possible thanks to the creation of a single mesh to represent the whole tree. We have designed a VRML world with four levels of detail of the tree (fig 10a). We apply force effects to the tree obtaining good visual results because there is no discontinuity over the branches (fig 10b).

Contour resolution	Polygons generated	
	Original tree	Interval refining
5	4998	9708
9	5028	10142
12	5343	13762
15	8435	24482

Table 1. *Comparison of the spatial cost of the reconstructed tree with the direct method and with refining at intervals for different contour resolutions*



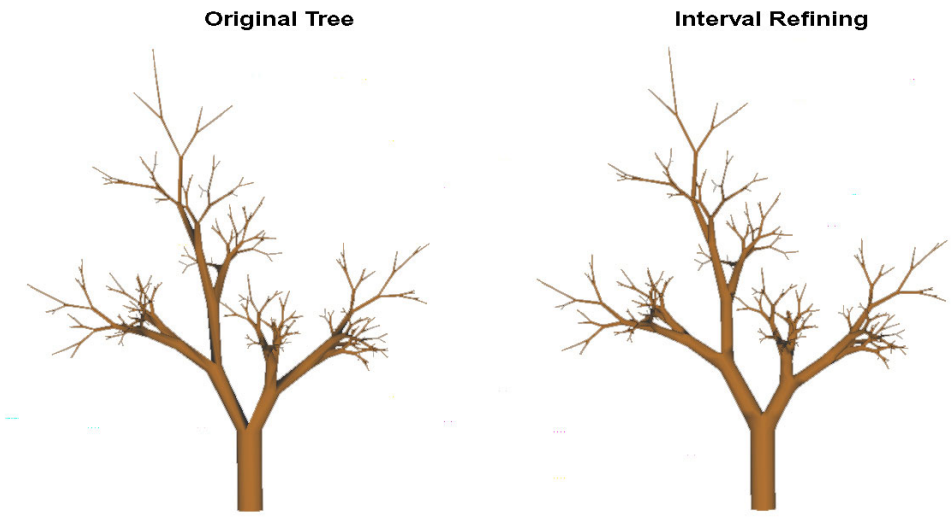


Figure 9. *Result of applying the developed algorithm to a chain generated by a RL-system. Shown on the left is the resulting 3D tree without the application of the method of refinement at intervals. On the right, the same tree with refined nodes is shown*

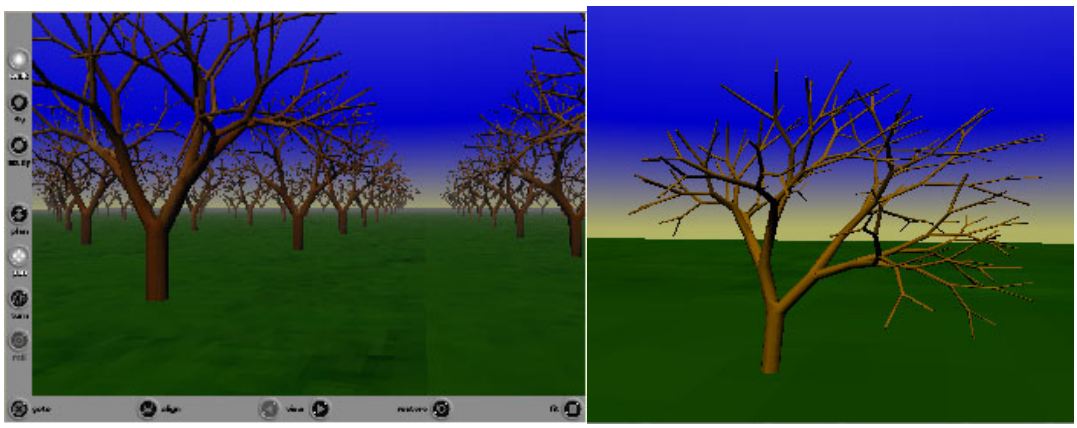


Figure 10. *Result of applying a decimation algorithm (a) and forces effects over the tree (b)*

5 Conclusions and Future Work

This article describes a new method to represent homogeneously a branched structure, for example a plant or tree. The representation of the model using a single polygonal mesh has the following advantages:

- It is easier to apply continuous textures over the whole model because we have a continuous mesh instead of a heterogeneous group of primitives.
- The problems of discontinuities and geometry superimposing are correctly solved.
- We can implement dynamic models on the mesh: force effects.

- It is possible to apply polygonal mesh simplification algorithms.

This method has been tested with trees and plants, and it can be applied to other branched structures like blood vessels and pipes.

The use of levels of detail through the application of simplification methods has the drawbacks of an increase in spatial costs and the appearance of discontinuities between the different LODs. It may produce undesirable visual effects when making the transition between two levels. Therefore, we propose as future work the implementation of a multiresolution model to support smooth transitions between LODs at a reasonable spatial cost. This model must take into account the branched nature of the objects represented. The decimation of the mesh must begin on the thinner branches. And the trunk must be decimated at the end of the process. It must avoid polygonal simplifications between disjoint branches. To improve the descriptive power of the method, the use of parametric curves is proposed to represent the contours and trajectories of the branches.

Bibliography

1. A. Lindenmayer: Matemathical models for cellular interaction in development, Journal of Theorical Biology, 1968, pp., 280-315
2. A. R. Smith: Plants, fractals, and formal languages. Proceedings of SIGGRAPH ’84 (Minneapolis, Minnesota, July 22–27, 1984) in *Computer Graphics*, 18, 3 (July 1984), pages 1–10, ACM SIGGRAPH, New York, 1984.
3. P Prusinkiewicz, Graphical applications of L-systems, Proceedings on Graphics Interface '86/Vision Interface '86, p.247-253, August 1986, Vancouver, British Columbia, Canada
4. P. Prusinkiewicz, A. Lindenmayer: The Algorithmic Beauty of Plants, New York, Ed. Springer-Verlag, 1990
5. P. Prusinkiewicz, M. James, R. Mech: Synthetic Topiary, Computer Graphics, 1994, pp., 351-358
6. R. Mech, P. Prusinkiewicz: Visual Models of Plants Interacting With Their Environment, ACM SIGGRAPH 96 Conference Proceedings, pp., 397-410

7. W. Reeves: Aproximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. Proceedings Siggraph'85, pp., 22-26.

8. J. Bloomenthal: Modeling the Mighty Maple. Proceedings of SIGGRAPH '85 (San Francisco, California, July 22-26, 1985), in *Computer Graphics*, 19, 3 (July 1985), pages 305–311, ACM SIGGRAPH, New York, 1985.

9. P. E. Oppenheimer: Real Time Design and Animation of Fractal Plants and Trees, Computer Graphics (SIGGRAPH '86 Proceedings), held in Dallas, Texas, August 18-22, 1986, vol. 20, pp., 55-64

10. P. de Reffye, C. Edilin, J. Françon, M. Jaeger, C. Puech: Plant Models Faithful to Botanical Structure and Development, Computer Graphics, 1988, vol. 22, no. 4, pp., 151-158

11. J. Weber and J. Penn: Creation and Rendering of Realistic Trees, ACM Siggraph 95, pp., 119-128.

12. B. Lintermann, O. Deussen: Interactive Modeling of Plants, IEEE Computer Graphics and Applications, 19,1 (Jan-Feb 99), 56-65.

13. J. LLuch, M. J. Vicent, R. Vivó, R. Quirós: GREEN: A new tool for modelling natural elements, WSCG'2000 International Conference on Computer Graphics and Visualization, Pilsen, Feb. 2000, Czech Rep.

14. D. House,G. Schmidt, S. Arvin, M. Kitagawa-DeLeon: Visualizing a real forest, IEEE Computer Graphics and Applications, 18,1 (Jan-Feb 98), 56-65.

15. N. Max: Hierarchical rendering of trees from pre-computed multi-layer Z-buffers. Rendering Techniques 96 pp 165-174. Springer-Wien

17. D. Meyers, S. Skinner, K. Sloan: Surfaces from contours, ACM Transactions on Graphics 11, July 1992, 228-258.

18. W.J. Schroeder, Jonathan A. Zarge, William E. Lorensen: Decimation of Triangle, Computer Graphics (SIGGRAPH '92 Proceedings), Vol. 26, No. 2, July 1992, pp. 65--70.