Interactive design of bonsai tree models

Frédéric Boudon, Premyslaw Prusinkiewicz, Pavol Federl, Christophe Godin, Radoslaw Karwowski

EUROGRAPHICS 2003 / P. Brunet and D. Fellner
(Guest Editors)

Volume 22 (*2003*), *Number 3*

Interactive design of bonsai tree models

Frédéric Boudon[†1] Przemyslaw Prusinkiewicz[‡] Pavol Federl[‡] Christophe Godin[†2] Radoslaw Karwowski[‡]

[1]INRA [2]INRIA
[†]UMR Botanique et Bioinformatique de l'Architecture des Plantes AMAP, Montpellier, France

[‡]Department of Computer Science, University of Calgary, Alberta, Canada

Abstract

Because of their complexity, plant models used in computer graphics are commonly created with procedural methods. A difficult problem is the user control of these models: a small number of parameters is insufficient to specify plant characteristics in detail, while large numbers of parameters are tedious to manipulate and difficult to comprehend. To address this problem, we propose a method for managing parameters involved in plant model manipulation. Specifically, we introduce decomposition graphs as multiscale representations of plant structures and present interactive tools for designing trees that operate on decomposition graphs. The supported operations include browsing of the parameter space, editing of generalized parameters (scalars, functions, and branching system silhouettes), and the definition of dependencies between parameters. We illustrate our method by creating models of bonsai trees.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques

1. Introduction

Plants are complex structures, consisting of multiple components. Consequently, plant models in computer graphics are commonly created using procedural methods, which generate intricate branching structures with a limited user input. Procedural plant models can be divided into two classes, *local-to-global* and *global-to-local* models [15]. In the local-to-global models, the user characterizes individual components (modules) of a plant, and the modeling algorithm integrates these components into a complete structure. This approach is particularly useful in the modeling and simulation of development for biological purposes. Due to the emergent character of the models, however, it is difficult to control the overall plant form. A notable exception is the modeling of topiary [13], which is based on simulating plant response to pruning. In the global-to-local models, in contrast, the user characterizes global aspects of plant form, such as its overall silhouette and the density of branch distribution. The modeling algorithm employs this information to infer details of the plant structure. The global-to-local approach provides a more direct and intuitive control of visually important aspects of plant form, and therefore is preferable in applications where visual output is of primary importance. These applications include the inference of plant structure from photographs [17] and interactive design of plant models, which is the topic of this paper.

The use of global information in plant model design can be traced to the work of Reeves and Blau [16]. In their method, the user specified a surface of revolution that defined the overall silhouette of a tree. The generative algorithm employed this information to infer the length of the first-order branches in the tree. The technique of Reeves and Blau was subsequently improved by Weber and Penn [19], Lintermann and Deussen [4, 10] and Prusinkiewicz *et al.* [15], who introduced numerical parameters and graphically-defined functions to control the density of branches, progression of branching angles, changes in the diameter and curvature of limbs, and other characteristics of the model.

An analysis of these previous approaches points to competing factors in selecting parameters (numerical, functional or compound, such as the entire plant envelope) that can be directly controlled. If the number of these parameters is small, the modeling algorithm must necessarily reuse some of them when generating different parts of the structure. This was already observed by Reeves and Blau, who wrote that higher-order branches had "many parameters inherited

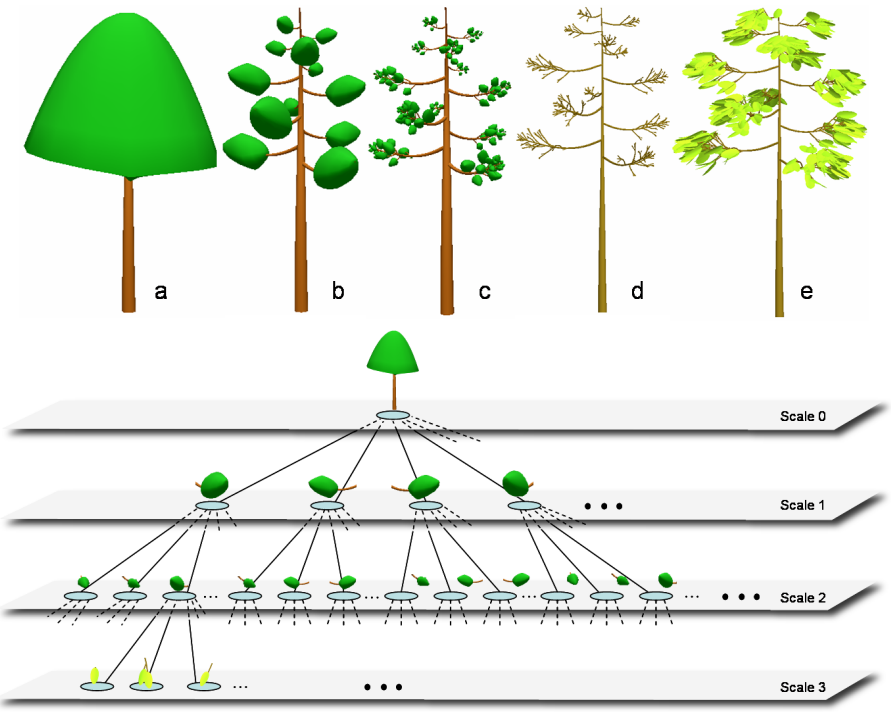

Figure 1: *Top: approximate representations of a tree structure at scales 0 to 3 (a–d), and the final tree model (e). Bottom: the corresponding decomposition graph.*

from the parent" in their model [16]. Judiciously reused parameters make it possible to effectively control models of highly repetitive structures, such as fern fronds, many inflorescences, and young trees [15]. Other plant models, however, may require direct control of individual plant components to capture their distinct features, creating a need for larger parameter sets. Unfortunately, interactive manipulation of these sets produces problems of its own: it is a tedious process in which the user is easily overwhelmed by the number of parameters and looses an intuitive grasp of their effects. Furthermore, having many parameters can make it more difficult to control the overall characteristics of the models. This is analogous to the interactive editing of curves and surfaces, where a large number of control points can make it difficult to control the overall geometry. A known solution to this problem is, of course, multiresolution editing, first introduced to geometric modeling by Forsey and Bartels [5], and subsequently generalized in different mathematical contexts (e.g., [18, 20]). In this paper, we extend the multiresolution modeling paradigm to the design of plant models.

A formalism for the multiresolution description of plants was introduced by Godin and Caraglio [6], under the name of *multiscale tree graphs* (MTG). We use it here in a simplified form, which we call decomposition graphs. A decomposition graph is a tree (in the graph-theoretic sense) that reflects the hierarchical structure of a plant induced by its branching order (Figure 1). Nodes of this graph are place-holders for the parameters that describe parts of the tree at different levels of the hierarchy, and thus at different levels of detail. In the process of interactive design of a plant model, the parameters describing higher-order branches are initially inherited from the parameters describing the plant as a whole. The user increases the diversity of the generated structure by breaking the pattern of parameter inheritance and editing parameters of selected components at a chosen level of the hierarchy. In such a way, the plant is gradually refined with a minimal expansion of the parameter set. The operations are effected using several software tools, which include *browsers* of the plant structure and *editors* of different parameters. A particularly important component is the *silhouette editor*, which makes it possible to directly manipulate three-dimensional, possibly asymmetric silhouettes of the branching systems.

The decomposition graph serves as the source of parameter values employed by the procedural model. We use global-to-local generative algorithms with the general structure described by Prusinkiewicz *et al.* [15], and implemented with the L-studio [14] modeling software. The L-system based modeling language L+C [8], extended with functions for accessing and manipulating the decomposition graph, makes it possible for the user to redefine or modify the generative algorithms if required by a particular model.

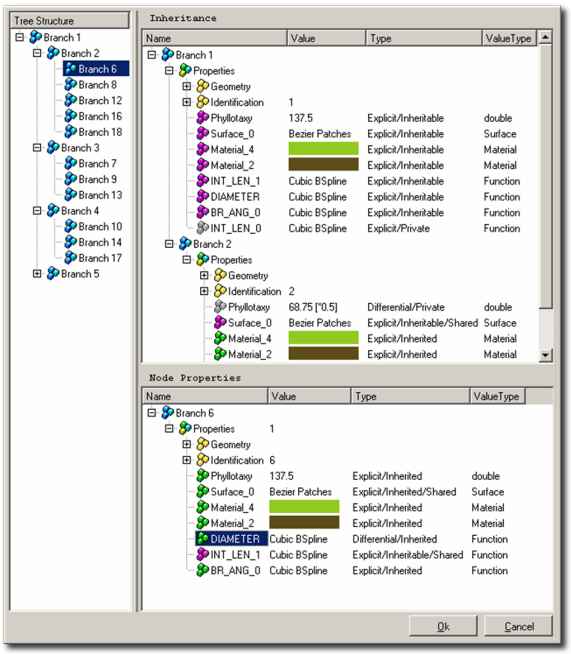

Figure 2: *The decomposition graph browser*

We illustrate our method by applying it to model bonsai trees. Real bonsai trees are often highly irregular, with the irregularities of the form inherent in biological development accentuated by human intervention. Consequently, bonsai models represent a challenging example of the need for flexible manipulation of plant shapes characterized by large numbers of parameters.

2. Browsing and editing plant structure

In our approach, a plant model is generated algorithmically, with parameters stored in the decomposition graph. As the number of nodes in the decomposition graph may be large, tools are needed to conveniently browse through this graph and access parameters associated with the individual nodes. We have developed two tools for this purpose: the *decomposition graph browser* and the *branching structure browser*.

2.1. The decomposition graph browser

The decomposition graph browser is manifested on the screen as a window with three panels (Figure 2). The left panel represents the hierarchical structure of the graph. It is visually similar to the file browsing tools in the Windows systems, and provides similar expansion/contraction operations to control which part of the graph is shown. This panel also makes it possible to select a specific node in the decomposition graph.

The attributes of the currently selected node are shown in the bottom right panel. A parameter is identified by its name, which provides a link to the generative program, and is further characterized by several fields. Among them, the *type* field specifies the inheritance status of the parameter, which in turn consists of up to three components. The first component determines whether a value is *explicitly defined* at a given node of the graph, *inherited* from another node, or *relative* with respect to the inherited value. The second component specifies whether the parameter value is *private* to the node, and thus cannot be inherited, or *public*, and thus inheritable. The third component indicates whether a parameter value is *shared* by several nodes that exist at the same level of the decomposition tree. The sharing mechanism applies only to parameter values that are defined explicitly (i.e., are not inherited). Aspects of the inheritance status are also visualized by assigning different colors to the icons associated with each parameter and node.

The distinction between private and public parameters affects the inheritance mechanism in the following manner. Consider the situation in which a particular parameter of a current node is inherited, the corresponding parameter of the parent node is private, and in the grand-parent node it is public. The parameter value in the current node will then be inherited from the grand-parent rather than the parent. More generally, the inherited parameter receives its value from the first node up the decomposition graph in which the corresponding parameter has been declared as public. By definition, all parameters in the root of the tree are public.

For any given parameter value in the model, the user needs to know from which node it originated. This information is available through the top right panel of the browser window, which shows parameters of all the nodes in the path from the root to the currently selected node. By inspecting which nodes are private or public, the user can identify the sources of parameter values inherited by the current node.

Definition and redefinition of the inheritance status of the nodes is an important aspect of the plant modeling process. Initially, all nodes inherit their parameter values from the nodes further up, along paths that eventually lead to the root of the decomposition graph. By accessing and editing an inherited parameter, the user creates its copy, and assigns it a new value. In this way, the number of independently controlled parameters increases, leading to a gradual diversification of the model components. With a menu, the user can also revert a parameter value to an inherited one, and, in general, change the inheritance status of any parameter. By carefully defining the inheritance structure of the decomposition graph, the user gradually constructs a parameter set that includes all the parameters required to capture the diversity of the modeled plant, but does not include superfluous parameters.

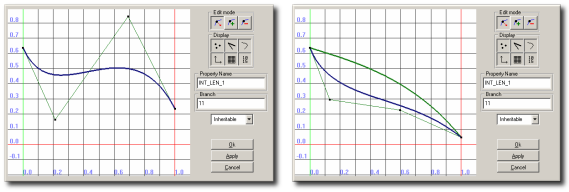

Figure 3: *2D function editor, in explicit (left) and relative (right) modes*

2.2. Parameter editors

Parameter values are modified using *editors*. An editor is opened by selecting a parameter in the panel showing the current node. The exact list of parameters, and therefore the editors, associated with the nodes depends on the underlying generative algorithm. In the simplest case of *scalar* values, the editor is a widget with a slider and an editable numerical field. More involved editors are used for compound parameters (attributes) of the node. These include a *material* editor for defining optical properties of the branch, editors of *curves* and *surfaces*, and a graphical *function editor* (Figure 3), as described in [14].

Some parameters (scalars and functions in the present implementation) can be declared as relative with respect to the inherited value. If this is the case, the value of the parameter is a combination of a value inherited from another node and a value defined locally. In the case of scalars, this means the actual parameter value is obtained by applying a locally defined offset (additive combination), or taking a locally defined fraction (multiplicative combination) of the inherited value. In the case of functions, the same combinations are achieved by taking the sum or product of the inherited and locally defined function. To facilitate the editing process, the function editor can show both the inherited and the modified function (Figure 3, right).

2.3. The branching structure browser

The branching structure browser (Figure 4) provides an alternative multiresolution view of the plant. It uses an iconic representation of the branching system to visualize a chosen level of the plant structure, and thus shows some of its geometric aspects, but does not explicitly show the inheritance relationships in the decomposition graph.

At the heart of the branching structure browser is the notion of the *branch silhouette*, which depicts the main axis and the outline (hull, envelope) of the branching systems contained within it. The browser arranges these silhouettes into a branching structure that conforms to the plant geometry at a user-selected scale. Thus, in addition to the silhouettes themselves, the browser visualizes the length of the internodes (segments of an axis between the insertion points of

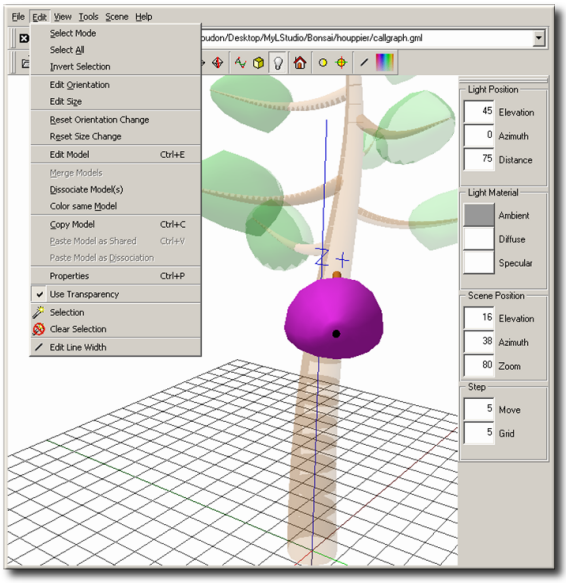

Figure 4: *A screenshot of the branching structure browser. The plant is represented at scale 2. The orientation of instance colored in purple is currently edited. The other instances become transparent, and give the user a focus on the current operation. The Edit menu displays all the possible editing operations.*

the consecutive branches) and the size and orientation of the branches (defined by the branching and phyllotactic angles).

The user can change the size and orientation of a branch by selecting and manipulating it using the mouse (for a general treatment of the interactive manipulation of branches see [12]). The user can also invoke an external parameter editor for the selected node. Most important in the context of multiscale editing is the silhouette editor, discussed in the next section. Used together, the branching structure browser and the silhouette editor provide a means for conveniently editing plant geometry in a manner that approaches direct manipulation.

2.4. The silhouette editor

The global geometry of a branching system is specified by its silhouette (Figure 5, see also Figure 1). The silhouette consists of a 3D curve, such as a polyline, a Bézier curve or a B-spline, which specifies the silhouette's *axis*. The silhouette also includes a potentially asymmetric *envelope* which represents the lengths of the lateral branches. Literature in botany contains a large variety of envelope models to represent the crowns of branching systems, for instance using cones, ellipsoids [11] or convex polyhedra [3]. We chose and implemented the envelope model proposed by Horn [7] and Koop [9], and later extended by Cescatti [1], which was designed

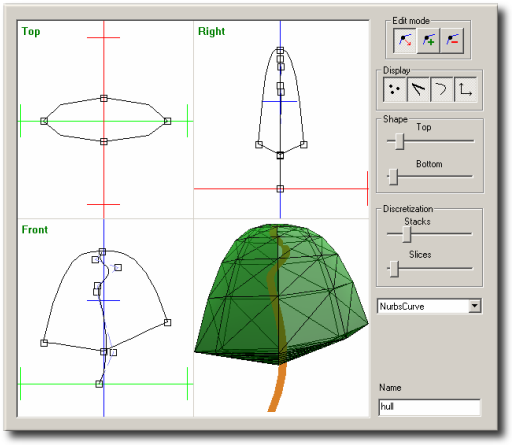

Figure 5: *Silhouette editor.*

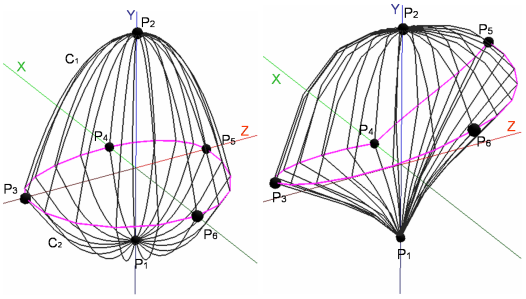

Figure 6: *Asymmetric envelopes defined by Cescatti to represent the crown shapes of trees. In the left figure, the control points P_3 to P_6 lie in the horizontal (xz) plane. In the right figure, points P_3, P_5 and P_6 have been moved vertically.*

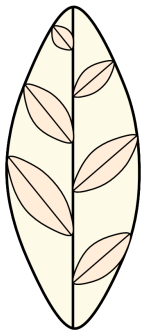

Figure 7: *Relationship between silhouettes at two different scales of plant hierarchy. The size of the silhouettes at the finer scale is determined by the shape of the silhouette at the coarser scale.*

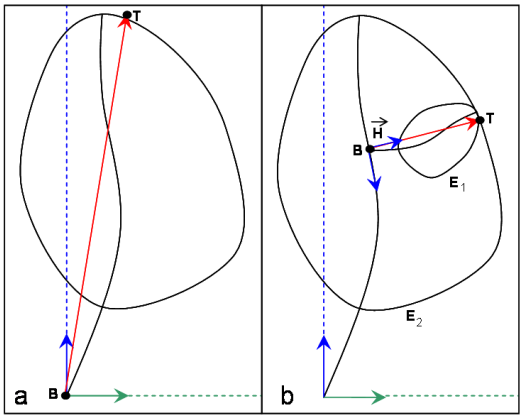

Figure 8: *Placement of a child silhouette E_c inside the parent silhouette E_p. a) The shape of both silhouettes. b) The result of placement.*

to flexibly represent a large variety of tree crowns in an intuitive fashion.

The Cescatti envelope is defined by six control points and two shape coefficients, C_1 and C_2 (Figure 6). The first two control points, P_1 and P_2, are the top and bottom points of the crown, respectively. The other four points, P_3 through P_6, describe a peripheral line at the greatest width of the crown when projected on the *xz*-plane. P_3 and P_5 are constrained to the *xy*-plane and P_4 and P_6 to the *yz*-plane. Finally, the shape coefficients describe the curvature of the crown above and below the peripheral line. Mathematical details of this model are described in the paper by Cescatti [1].

3. Multiscale constraints

Parameters associated with different nodes of the decomposition graph may be related to each other not only by the inheritance pattern, but also by their meaning. An example is the relationship between the shape of a silhouette of a branching systems and the sizes of the silhouettes associated with the lateral branches (Figure 7). Clearly, one cannot modify the overall silhouette without affecting the size of the individual branches, and vice versa. In general, the information in a parent node of the decomposition graph is related to that in the child nodes, because both the parent and the children describe the same branching system. Since this relation spans different scales of plant description, we call it a *multiscale constraint*.

Multiscale constraints can be satisfied in a bottom-up, local-to-global fashion, or in top-down, global-to-local fashion. These terms describe the direction in which the constraint information propagates in the decomposition tree. The top-down direction is better suited for the interactive plant design, which commonly begins with the overall plant silhouette, and proceeds by gradually refining it [15] (c.f. Section 1).

At a practical level, we thus face the problem of placing a child silhouette E_c inside the parent silhouette E_p. To this end, we add an extra control point T to the description of the silhouette, as shown in Figure 8. The vector $\overrightarrow{BT}$ that connects the base of the silhouette E_c to the point T is used to orient this silhouette in space and determine its size. First, point B is positioned at the branching point specified by the generative algorithm. Next, the vector $\overrightarrow{BT}$ is aligned with the branch direction given by the branching and phyllotactic angles. Finally, the silhouette E_c is scaled so as to place point T on the parent silhouette. A reference frame associated with the child silhouette E_c can optionally be used to rotate it around its own axis.

The multiscale constraint discussed above relates parameters associated with different nodes in the decomposition graph, but does not affect its structure (topology). A more complicated situation occurs when the user manipulates the density of branch distribution along an axis. The density function associated with a parent node determines the number of the child nodes, and therefore affects the structure of the decomposition graph. According to our approach, this structure is generated algorithmically, which means that the generative algorithm must be re-run to satisfy the branch density constraints. The coupling between the generative algorithm and the interactive manipulation of parameters is schematically depicted in Figure 9, and discussed in more detail in the next section.

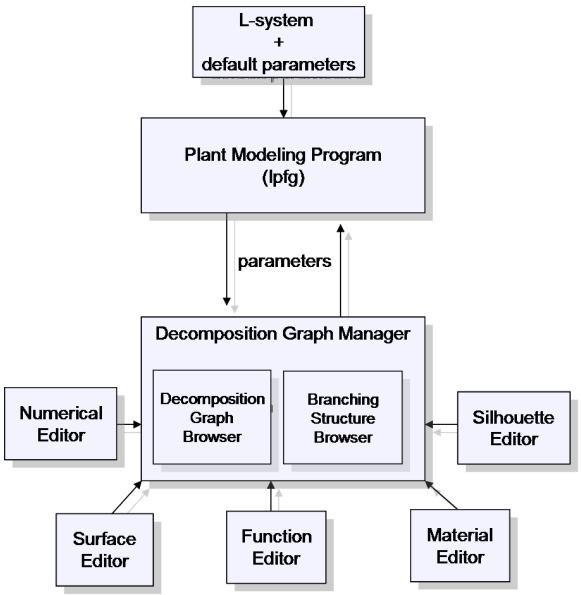

Figure 9: *Interaction of various components during the modeling process.*

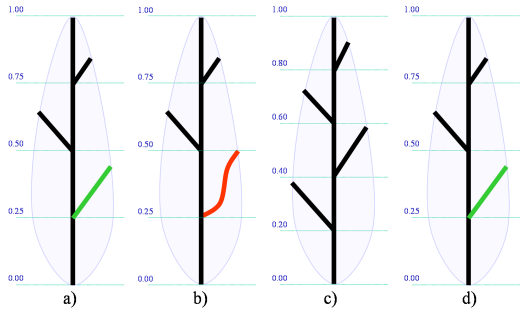

Figure 10: *Effect of decoupling. a) Initial plant structure, showing the default shape of the bottom branch. b) The shape of the bottom branch (position 0.25) was manually adjusted. c) The branch density was increased. d) The branch density is restored to the original value, and all branches are straight lines.*

4. The modeling process

The user begins the modeling process by specifying a generative algorithm in the L-system-based language L+C [8]. (Our models are constructed using the global-to-local paradigm, and thus are more properly described by Chomsky grammars than L-systems [15]. Nevertheless, we continue to use the term "L-systems", because L-system and Chomsky productions can be combined seamlessly in the same model, making clear separation difficult). During its first execution, this algorithm makes calls to functions that create the decomposition graph and define parameters for some nodes. The nodes for which the parameter values have not been explicitly defined inherit their values from the parent nodes, as described in Section 2.1. Specifically, if the initial execution of the algorithm defines parameter values for the root only, all nodes of the decomposition graph will share the same set of values.

Once the initial decomposition graph has been created, the parameters contained within it can be interactively edited. Following that, the generative algorithm must be re-run to reconstruct the plant structure. In principle, the algorithm then accesses the values stored in the decomposition graph. To associate the nodes of the graph with the specific branches of the generated structure, the branches and the nodes are identified by their *paths* to the top of the decomposition tree. A path of a node is recursively defined by three components:

- the *path* of the node's parent;
- the normalized *position* of the branch along the axis;
- a *number* identifying the branch, if several branches are attached to the same point of their supporting axis.

Unfortunately, storing an algorithm's parameters in the decomposition tree may lead to problems. As a result of parameter manipulation, the paths assigned to the branches during the re-execution of the generative algorithm may dif-

fer from the paths stored in the decomposition graph. This will occur, for example, if the user has changed the function that defines the density of branch distribution along an axis (c.f Section 3). In the case of such a decoupling, the generative algorithm removes the nodes that are no longer used, and adds new nodes to the decomposition graph for those branches that do not have a corresponding node. These adjustments may have unintuitive consequences from the user's perspective. Consider the example illustrated in Figure 10. The branch density function for the initial plant structure (a) determines the main axis will have three lateral branches. The normalized positions of these lateral branches are 0.25, 0.5 and 0.75, respectively. By default, all axes are straight. Now, suppose the user changes the shape of the bottom branch to a curved one, as illustrated in Figure (b). Next, the user changes the branch density of the main axis, increasing the number of lateral branches to four. When the algorithm regenerates the plant structure, it attempts to obtain parameter values for the branches whose normalized positions are 0.2, 0.4, 0.6 and 0.8. These paths, however, do not correspond to any of the existing nodes in the decomposition graph. Consequently, new nodes are created for all the lateral branches, while the old ones are removed from the graph. The new nodes are assigned default parameter values, which results in the structure shown in Figure (c). The shape of the axis associated with node 0.25 is now permanently lost. Thus, even if the branch density is later returned to the original value, the algorithm will not restore the bottom branch to its curved shape (Figure d).

The problem described above can be attributed to the fact that the management of parameters is decoupled from the algorithm that uses them to construct the plant. We perceive this problem as a very fundamental one: in order to interact with the plant, we personalize each branch so that we can select and modify it. Unfortunately, there is no robust method for maintaining the identity of branches during modifications that may displace them, of even temporarily remove them from the structure. In practice, we reduce the impact of this problem by first defining the distribution of the branches, then modifying their shape from the default.

5. Results

We applied our method to model a number of bonsai trees. They present a challenging modeling problem because of their highly irregular structures. While real bonsai trees are a result of interplay between biological development and human intervention, our models are the result of interplay between the biologically-based generative algorithms and interactive manipulation.

The results are shown in Figures 11 to 15. For reference, we also show some of the real plants we attempted to model. Each model was created in approximately 3 hours.

On a PC with a 1 GHz Pentium III processor, the process

Figure 11: *Bonsai 1 : bunjinji style, photograph from* [2]

of generating the detailed models of bonsai shown in Figures 11, 13, 14 and 15 takes between 1 and 2.5 seconds. The models in Figure 12 were the longest to generate (10 and 12.5 seconds) due to the large number of needles (modeled as generalized cylinders).

6. Conclusions

We have presented an approach for modeling plants based on a global-to-local design methodology, consistent with artistic techniques. To this end, we formalized a multiscale model of a plant by defining a decomposition tree, the nodes of which represent specific branches of the plant structure. The parameters needed to construct the plant are then associated with the nodes of the decomposition tree. We proposed inheritance and parameter sharing as a method for minimizing the total number of parameters needed, while giving the user the opportunity to refine any aspect of the model. Our ap-

Figure 12: *Bonsai 2: nejikan style (twisted cascade) and Bonsai 3: fukinagashi style (windswept)*

Figure 13: *Bonsai 4: chokkan style (formal upright), photograph from* [2]

Figure 14: *Bonsai 5: sankan style, with three branches originating at the same point*

Figure 15: *Bonsai 6: kengai style (formal cascade), photograph from* [2]

proach alleviates the difficulty of managing and navigating through a complex parameter space, which is an issue in interactive plant design [4, 10]. We also observed the impact of multiscale constraints on the modeling process.

At a practical level, we have implemented a system based on the above paradigms. It consists of tools that allow the user to select a branching structure at any level of the hierarchical plant organization, and interactively edit its parameters. We found that these tools make it possible to design plant models relatively quickly and in an intuitive manner. Finally, we have demonstrated the usefulness of our system by modeling several bonsai trees.

There are a number of areas where our results can be further improved. We believe the issue of attributes being decoupled from the procedural algorithm deserves more examination. For example, we could associate nodes in the decomposition graph with ranges of branch positions, rather than single position, thus potentially reducing the decoupling artifacts discussed in Section 4 (Figure 10).

The attribute inheritance mechanism we have considered in this paper only relates branches at different scales. This approach is well suited for modeling monopodial plants,

with a clear distinction between the parent axis and its laterals. However, in sympodial plants a branch may support another branch at the same scale [6]. To facilitate modeling of sympodial plants, our inheritance mechanism should be extended to within-scale relationships between the nodes.

Finally, the visual quality of our models could be improved by adding more details using displacement mapping.

Acknowledgments

We would like to thank Christophe Pradal for his precious help, Christophe Nouguier for his first implementation of the GEOM library which provided good support for our work, Frank Perbet, Lars Mündermann and Brendan Lane for their explanations and advice, Jennifer Walker for her editorial help, and all the people from the University of Calgary Graphics Jungle Laboratory for creating such a friendly working environment. This work was supported by the France-Canada Research Foundation, Natural Sciences and Engineering Research Council of Canada, and Institut National de la Recherche Agronomique, France.

References

1. A. Cescatti. Modelling the radiative transfer in discontinuous canopies of asymmetric crowns. I. Model structure and algorithms. *Ecological Modeling*. 101(2-3):263–274, 1997. 4, 5
2. R. R. Clark and P. Voynovich. Outstanding american bonsai: A photographic essay on the works of fifty american bonsai artists. Timber Press, September 1989. 7, 8
3. C. Cluzeau, J. L. Dupouey, and B. Courbaud. Polyhedral representation of crown shape. A geometric tool for growth modelling. *Annals of Forest Science*. 52:297–306, 1995. 4
4. O. Deussen and B. Lintermann. A modelling method and user interface for creating plants. In *Proceedings of Graphics Interface '97*, pages 189–197, 1997. 1, 8
5. D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. Proceedings of SIGGRAPH 1988 (Atlanta, Georgia, August 1-5, 1988) in *Computer Graphics*, 22, 4 (August 1988), pages 205–212, ACM SIGGRAPH, New York, 1988. 2
6. C. Godin and Y. Caraglio. A multiscale model of plant topological structures. *Journal of Theoretical Biology*, 191:1–46, 1998. 2, 9
7. H. S. Horn. The adaptive geometry of trees. Princeton University Press. Princeton, N.J. 1971. 4
8. R. Karwowski. Improving the process of plant modeling: The L+C modeling language. Ph.D. thesis, University of Calgary, October 2002. 2, 6
9. H. Koop. SILVI-STAR: A comprehensive monitoring system. *Forest Dynamics*. Springer, Berlin, pp. 229, 1989. 4
10. B. Lintermann and O. Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, 1999. 1, 8
11. J. M. Norman and J. L. Welles. Radiative transfer in an array of canopies. *Agronomy Journal*. 75:481–488, 1983. 4
12. J. Power, A. J. Bernheim-Brush, P. Prusinkiewicz, and D. Salesin. Interactive arrangement of botanical L-system models, 1999. Proceedings of the 1999 ACM Symposium on Interactive 3D Graphics (Atlanta, Georgia, April 26–28, 1999), pp. 175–182. 4
13. P. Prusinkiewicz, M. James, and R. Měch. Synthetic topiary. Proceedings of SIGGRAPH 1994 (Orlando, Florida, July 24–28, 1994). ACM SIGGRAPH, New York, 2001, pp. 351–358. 1
14. P. Prusinkiewicz, R. Karwowski, R. Měch, and J. Hanan. L-studio/cpfg: A software system for modeling plants. In M. Nagl, A. Schürr, and M. Münch, editors, *Applications of graph transformation with industrial relevance*, Lecture Notes in Computer Science 1779, pages 457–464. Springer-Verlag, Berlin, 2000. 2, 4
15. P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. Proceedings of SIGGRAPH 2001 (Los Angeles, California, August 12–17, 2001). ACM SIGGRAPH, New York, 2001, pp. 289–300. 1, 2, 5, 6
16. W. T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. Proceedings of SIGGRAPH '85 (San Francisco, California, July 22-26, 1985), in *Computer Graphics*, 19, 3 (July 1985), pages 313–322, ACM SIGGRAPH, New York, 1985. 1, 2
17. I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller. Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21(3):53–61, 2001. 1
18. E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for computer graphics: theory and applications*. Morgan Kaufman, San Francisco, CA, 1996. 2
19. J. Weber and J. Penn. Creation and rendering of realistic trees. Proceedings of SIGGRAPH '95 (Los Angeles, California, August 6–11, 1995). ACM SIGGRAPH, New York, 1995, pp. 119–128. 1
20. D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. Proceedings of SIGGRAPH 1997 (Los Angeles, California, August 3–8, 1997), pp. 259–268. 2