A Modelling Method and User Interface for Creating Plants

Bernd Lintermann[†‡] and Oliver Deussen[§]

Abstract
We present a modelling method and graphical user interface for the creation of natural branching structures such as plants. Structural and geometric information is encapsulated in objects that are combined to form a description of the model. The model is represented graphically as an structure graph and can be edited interactively. Global and partial constraint techniques are integrated on the basis of tropisms, free-form deformations and pruning operations to allow the modelling of specific shapes. We show examples to illustrate the design process and evaluate the user interface.

---

1. Motivation

The design of natural objects such as trees, bushes or flowers is a challenging task in two respects. One is the sometimes enormous structural complexity, the other is the large amount of geometrical properties that must be taken into consideration.

Both tasks require their own optimized modelling techniques. To combine adequate modelling methods for geometrical and structural properties in a single system, we describe models by an object (or component) hierarchy. Components contain both geometrical and structural data together with corresponding methods.

To define a model, icons (graphical representations of components) are combined to a structure graph. This graph represents creation rules and defines a context-free rule system.

The geometric properties are handled as parameters of the components. Every object is able to create geometry. A basic parameter set together with manipulation methods is common to all components.

We first give an overview of recent work on plant generation. Next, we introduce our way of modelling and the corresponding graphical user interface.

The creation of a dandelion (ref. Figure 1) will serve

† Department of Computer Science, University of Karlsruhe, P.O.Box 3980, D-76128 Karlsruhe, Germany

‡ ZKM-Karlsruhe, Lorenzstr. 19, D-76135 Karlsruhe, linter@zkm.de

§ Department of Computer Science, Otto-von-Guericke University of Magdeburg, P.O.Box 4120, D-39016 Magdeburg, Germany, deussen@isg.cs.uni-magdeburg.de

as an example for using the method including geometrical and structural modelling. Finally, practical aspects are discussed and experience with trained and novice users are documented.
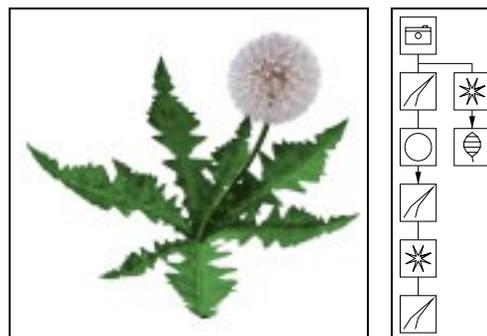

Figure 1: *Dandelion and corresponding structure graph.*

Some geometric aspects of the modelling process are discussed in[1]. The present article deals mainly with the user interface and the modelling method itself, a shorter version of this paper is given in[2].

2. Recent Work

Various publications have appeared in the last few years concerning the problem of plant generation. In general, two approaches can be distinguished. First, formal techniques have been developed for mastering the structural complexity of branching structures. Second, more model-oriented approaches have been introduced which concentrate on the analysis and generation of special plants and their geometry.

Among the formal methods, rule-based approaches and especially L-systems[3] have been extensively studied. L-systems are string rewriting systems similar to Chomsky grammars. A sequence of letters is derived from a starting word, called the axiom, by parallel application of string rewriting rules. In a second step the generated string serves as a command sequence for a turtle graphics interpreter that generates geometric data.

This method is explained in[4]. Extensions such as context-sensitive and parametric grammars as well as stochastic application of rewriting rules are presented. These techniques allow the simulation of physiological mechanisms regulating the growth process such as hormone secretion. Special properties of plants, e.g. photo- and gravitropism, can be modelled[5]. Another extension to L-systems, called dL-Systems[6], define additional differential equations to introduce continuity to the so far discrete approach. Most of the work concerning L-Systems is based on textual representation and manipulation and geometric and structural aspects are handled by the same overall method.

Other models deal only with trees: De Reffye et al.[7] developed a procedural model based on birth and death of growing buds. A dynamic process generates the geometric data. Holton proposed a vascular model[8]. A strand is assigned to any path from the root of a tree to its leaves. The number of strands in a fork determines the fork angle, length and taper of branches.

A generic tree model is presented by Weber and Penn[9]. The (textual) variation of a given parameter set enables even a non-biologist to generate several natural looking trees. The generation of more general objects by parameter variation is described in[10].

Generic tree models are also used in AMAP[11], a system for the generation of landscapes which offers a library of algorithms for generating several plants. Here, parts of the problem can be interactively manipulated with graphical user interfaces. This is essential for modelling purposes that require a huge number of parameter changes, as stated by Oppenheimer[12].

TreeMaker[13], a software for the generation of trees, offers also a graphical user interface. A tree can be generated interactively by changing structural parameters. Several graphical actions like cutting a stem are possible. The system is limited to trees with a specific structure.

3. The modelling method

Our intention was to find a general description of natural branching objects. A set of components was created containing geometric as well as structural information. The components are graphically represented by icons. These icons are combined to form a structure graph of the model (c.f. Figure 1).

The user selects icons out of a toolbox (c.f. Sect.3.1) and combines them by hanging them on a root icon. The process is shown in Figure 2. By double clicking an icon,
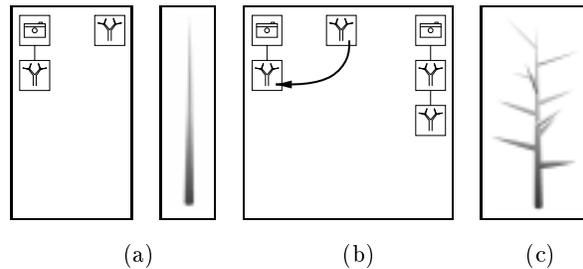

(a)   (b)   (c)

Figure 2: *Graphically interactive modelling: (a) editing window with a simple structure graph after selecting a new component; (b) the component is added to the structure graph by moving the icon upon the desired parent icon (branching combination); (c) resulting structure graph and geometrical result.*

the parameter set of the corresponding component is displayed and can be edited.

The generation of the geometric data according to the structure description is done as follows: The component of the camera icon defines the view and creates the corresponding components for all child icons. The components are forced to generate their geometrical output and to create subsequent components. This is done until the whole structure graph is traversed.

In contrast to other geometry-describing data structures like PHIGS+ and Inventor[14, 15] we use procedural modelling and describe creation rules for components which also include recursion and algorithmic creation of components. The latter is done by specialized components and will be described below.

	Simple: A component that produces a simple geometry (cube, sphere, cylinder etc.).
	Revo: Produces a surface of revolution. Definition is done by editing a polygon which describes the silhouette.
	Horn: A Sweep component that places other components on a user-defined curve. It is used for stems, twists, etc.
	Leaf: Is used for the construction of natural leaves. The leaf is defined by its outline, several curvatures (see Figure 7) and its material. Textures can be used to define the inner structure.

Table 1: *Components for generating geometry*

3.1. Component types

All components, as mentioned above, offer a basic functionality. They can produce geometric primitives and define material properties like color and texture, the recursion depth and transformation parameters for the com-

Tree: Basic component for trees, creates the geometry of a stem and multiplies subsequent components as branches. Parameters are the distribution of branches, their scale, angle etc.

Hydra: Multiplies subsequent components on a circle with uniform angles and direction perpendicular to the direction of the parent component.

Wreath:Multiplies subsequent components similar to hydra, but with direction parallel to the direction of the parent component.

PhiBall: Multiplies subsequent components on a section of a sphere according to the golden section (details see[1])

Table 2: *Iteration and arrangement of components*

World: Introduces light and gravitational fields to subsequent Tree components.

Pruning: Defines a pruning operation on the geometry generated by subsequent components.

FFD: Enables free form deformations of the geometry produced by subsequent components and is also used to switch previous FFDs off. Three definition methods can be used.

HyperPatch:Enables free form deformations of the geometry produced by subsequent components and is also used to switch previous FFDs off. The deformation is modelled by moving control points of a 3-D patch.

Table 3: *Introduction of constraints, geometrical Transformations*

ponent itself and all geometric output. Table 1 shows the geometry producing components that are derived from the simple component which represents the basic functionality.

Some components multiply others algorithmically (Table 2), some can be used for defining global and partial constraints (Table 3).

Practical experience showed us that this set sufficiently produces a broad variety of plant models (see below). In the future it is intended to add some components realizing other creation algorithms.

3.2. Combining Components

Different mechanisms for generating geometry including recursion are selected by several ways in combining components. First, a component can be a child of another which means that if the parent is created it produces its geometry and forces the child to generate its geometry

relative to its own. This is the normal way of combination and is encoded by moving one icon upon another using the left mouse button (c.f. Figure 3(a)).

Another possibility is to generate a component as a branch or rib of another component. This was done in Figure 2 to generate the twists of the tree. This kind of combination can be selected if special components like the Tree or Horn component are to be combined.
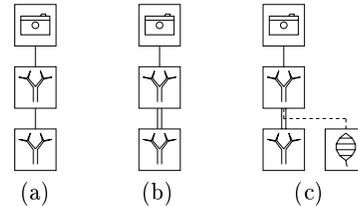

Figure 3: *Methods in combining icons: a) child/ branch/ rib combination; b) recursive combination; c) recursive combination with leaf link.*

If a component is a child of a multiplying component like Hydra, Wreath or PhiBall, the link represents as many components as have been multiplied. The parameters for these components are changed by special parameters of the multiplying component or by functions (c.f. Section 3.4) that use random values or the iteration number, which is also supplied by the multiplying component.

3.2.1. Recursive combination

A recursive combination is created if an icon is selected by ´using the middle mouse button and moved onto one of its child icons. In this case another icon representing the same component is drawn and the path between the two icons is displayed by a double line to indicate the recursion (Fig. 3(b)). If the user selects one of these icons, all other icons representing the same component are also highlighted. This enables us to represent the graphical description, which is a directed graph, visually by a tree.

One of the basic parameters of each component is the recursion depth. By selecting an appropriate value the recursion can be modelled. After the last recursive creation it is possible to specify another component which is created as leaf (leaf in the sense of the structural description). This is graphically encoded by moving an icon (using the right mouse button) upon one of the icons of a recursive defined component. The combination is represented visually as a dashed line (Fig. 3(c)).

As mentioned above, the structure graph defines a rule system. A parent-child dependency can be seen as a rule which is applied after the parents creation. Because no other rules or components influence the application of the rule the system is context-free. Besides serving us for representing our models, the method may be used for defining arbitrary context-free rule systems and especially L-Systems.

3.3. An example

The modelling process can be explained best by giving a short example. We shall explain the creation of a dandelion which gives a good overview on the functionality of the system.

At first a Horn component is selected. It will serve as one hair of the tiny umbrellas of the dandelions blossom. A slight curvature is established by the transformation parameters of the horn (Fig. 4(a)).
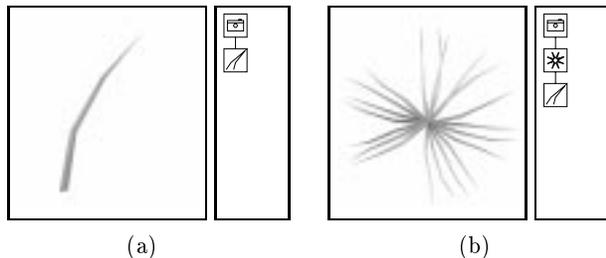

(a)   (b)

Figure 4: *Creation of a dandelion: (a) one hair; (b) the head of an umbrella.*

Next a Hydra component is used to multiply the Horn. The Horn is hung under the Hydra (by moving it upon the Hydra icon) and afterwards Horn and Hydra are hung under the Camera. The transformation parameters of the Hydra are used to obtain the orientation of the hair as can be seen in Figures 4(b) and 5(a).
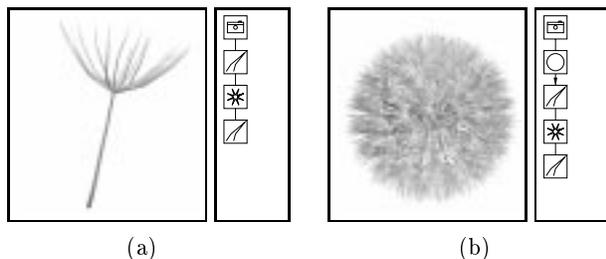

(a)   (b)

Figure 5: *(a) The whole umbrella. (b) The head is made by arranging umbrellas on a sphere according to the golden section.*

Now we create the stem of the umbrella. Again a Horn component is used for this (the resulting structure graph is shown in Figure 5(a)). Next a PhiBall component is used for multiplying the umbrellas on a sphere. Everything is hung under the PhiBall.

In Figure 5(b) a little arrow between the PhiBall component and the rest of the structure graph is displayed. This arrow indicates that the geometry produced by the subtree is calculated once and later only referenced for displaying by the PhiBall. This is much faster than the normal way of geometry generation where each component created by the PhiBall calculates its own geometry. In that case the geometry can vary from component to
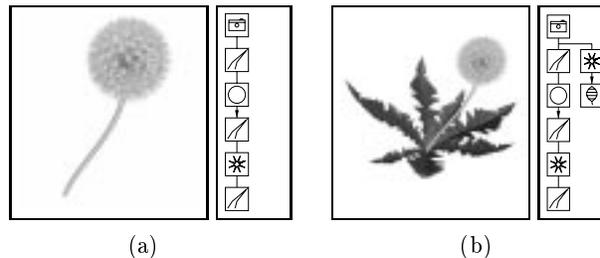

(a)   (b)

Figure 6: *(a) Modelling the stem; (b) adding the leaves.*

component. This is not necessary here and by this the direct geometry generation is enabled for the umbrella.

Next a Horn component is used for the main stem. The curvature is introduced again by the basic parameters of the Horn. A spline can be used for modelling the thickness of the stem along the main axis.
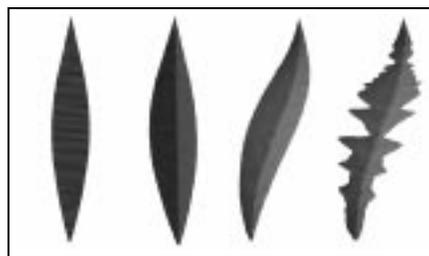

Figure 7: *Modelling leaves is done by introduction of axial and lateral curvature and editing the outline.*

Now we create the leaves and use the Leaf component for creating the geometry. Again a spline is used to generate the jagged outline of the leaves. The curvature is modelled in a similar way (Figure 7).

A Hydra component is used to multiply the leaves and both Hydra and the main stem are hung on the Camera icon. The resulting structure graph is shown in Fig. 6(b).

3.4. Functional modelling

In the upper example one may argue that the leaves of a real dandelion never look all the same – and in Figure 6(b) they really do not. There is a general way to introduce randomness at many places of the system. Wherever a parameter intervall is to be defined (e.g. the rotation along the main axis of a Horn, the curvature of a leaf) a function can be applied before the system uses a parameter of that interval.

The modeller offers standard mathematical functions like sine, cosine, etc., but also a random function. Parameters like recursion depth or iteration number can be used inside these functions. The user can define arbitrary functions for each parameter interval.

To demonstrate the effect an agave is shown in Figure 8. The more vertical leaves are less curved than the rest. This was done by using the iteration number which

Figure 8: *Agave with leaf curvature in dependency to the iteration number and a random function.*

was set by the PhiBall component during multiplying. By adding some noise a natural outline can be achieved.

3.5. Introduction of tropisms

Often a whole model or parts of it are influenced by global constraints like gravitation or light direction. To introduce such parameters a World component is used.

The fields that are defined here have an influence on all subsequent Tree components. These components have special parameters specifying how sensitive - along the main axis of the stem - the tree is to gravitational (gravitropism) and light fields (phototropism).
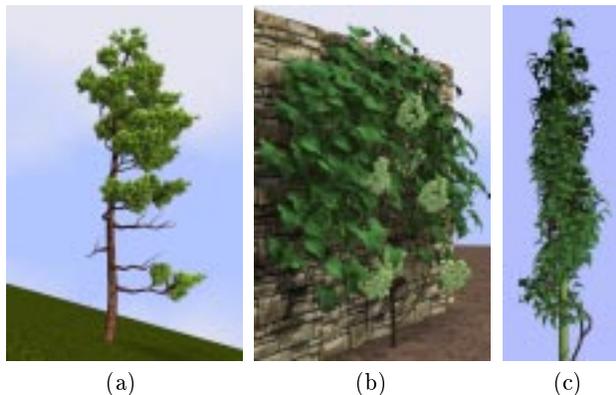

Figure 9: *Horizontal tropisms simulate the influence of wind (a) and force the wine to grow along a wall (b). A circular tropism is used for modelling an ivy around a stick (c).*

Three examples of tropisms in different applications can be seen in Figure 9. Tropisms can play the role of constraints as can be seen in Figure 9(b). By specifying the regions where growth has to take place, the plant achieves a specific shape.

For the moment the fields have to be defined functionally, but in the future it is intended to use a kind of

voxel model as done in[16], where the user selects regions of preferential growth.

3.6. Free form deformation

Another way to change the geometry of the entire model or just of parts of it is to use free form deformations. Two component types are designed for that purpose. The FFD component allows the user to define the deformation separately for each axis, whereas the 3-D Patch component uses a 3-D cube where points are to be moved.

Figure 10: *Selective free form deformation applied to a pine: (a) undeformed model, (b) model affected by a FFD of the twists, the needles remain unchanged.*

The user selects one or more points and moves them parallel to the viewing plane. The points are control points of a 3-D Bézier function which defines the desired deformation.

In Figure 11(a) the three dimensional representation of the 3-D Patch can be seen. Whereas the FFD component is useful for deformations that are to be defined exactly, the 3-D patch is used for more intuitive deformations.
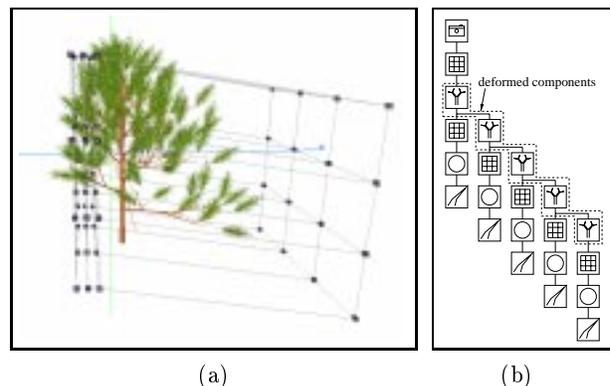

Figure 11: *(a) User interface of the 3-D Patch component; (b) Structure graph of the pine.*

Often it is required to deform only parts of the model. For example the twists of a tree should be deformed, but not the leaves or needles. This can be done by placing another FFD or 3-D Patch component in the subtree affected by a free form deformation. This component is used as a separator for the deformations defined before.

Figure 12: *Pruning an element of a hedge: a) Boxed, b) spherical, c) conical, d) cylindrical*

An example of partial free form deformation can be seen in Figure 10(a). The twists of the pine are deformed, but the needles remain unchanged. In Figure 11(b) the structure graph is shown with all deformed components marked.

3.7. Modelling of exceptions

Sometimes one may change properties of single components out of a set of multiplied ones. For example the user might to delete some of the seeds of a sunflower or break one twist of a tree.

The general way of modelling such effects is to define components that model exceptions. Each multiplying component holds a list of flags which indicate if the component shall generate geometry for a certain iteration or not. The flag is switched on for the iteration which is the exception and switched off for all other iterations. This is done vice versa for the component that defines the normal geometry.

Another possibility is to create two identical components which differ only in the distribution of successors. The distribution of one component generates the normal geometry, the other distribution modells the exeption. An example of this feature can be seen in Figure 9(a) where the dead twists of the pine were modelled that way.

3.8. Pruning

Often the outline of plants is modelled by humans. Typical examples are cutting operations applied to trees or pruning of hedges. While the former can be modelled by exceptions as shown above, the latter requires a new methodology.

An additional pruning component is introduced that modifies the generated geometry of subsequent tree components in the structure graph. The pruning component has two kinds of children, one kind defines the geometry to be pruned, the other defines a volume.

This volume is used to define the boundary of the geometry to be generated. The geometry can be forced to take place within or outside of the volume. If more than one volume defining component is given, the user is able to apply CSG operations on the volumes like creating the intersection or the union.

If the geometry generation process encounters the border of the volume, three types of reactions can be forced:
- **Cutting**: The geometry (the stem or twig) is cutted as if a real cutting operation was applied.
- **Cutting and reshape**: The stem or twig is also cutted, but its outline is changed in a way that it looks like it has grown naturally until it reached the boundary.
- **Scaling**: The outline ot the twig is changed as above, but now subsequent branching twigs are also affected. By using this method, the full outline of the tree looks natural and fits into the pruning volume.

In Figure 12 a little bush was pruned by using different pruning objects. Here, the tiny twigs were simply cutted to achieve the outlook of a hedge.

4. Reducing the displayed complexity

After showing the modelling technique the last part of the paper deals with practical aspects. Very important for the efficiency of the system is the amount of geometry to be handled during plant generation. As shown in Section 3.3 it is possible to reduce the computational cost for generating geometry. But often the number of displayed triangles is more critical. For example the dandelion of Figure 1 has about 147,000 triangles if displayed with full complexity.

Figure	Triangles	Generation	Display
1	2743/147991	0.22	3.43
8(a)	19510/26590	0.78	1.23
9(a)	36561/36561	5.33	0.78
9(b)	37075/154003	4.81	3.83
10	39624/39624	5.09	0.86
15	22753/41713	1.55	0.91

Table 4: *Model complexity and display time on an SGI Indigo 2 Extreme (134 MHz) in sec.*

Table 4 shows the maximum time consumed by the system for generating the internal data structure. The right column gives the time needed for displaying the data. The geometrical complexity is indicated by

the number of generated/displayed triangles. Depending on the structural model complexity, between 6,000 and 50,000 triangles per second can be generated by an Indigo 2 Extreme (134 MHz).

The values given in Table 4 are maximum time limits for generation and display. The user usually edits only parts of a model with low complexity. This can be done by disconnecting other stuff from the camera.

Another method is "hiding" components. If a component is hided, it generates no geometrical data but creates other components in the normal way. The combination of both techniques allows to handle medium-sized models on high-end PCs using LINUX with a soft implementation of Open GL.

5. Evaluation of the interface

Modelling technique and the interface were tested by experienced and novice users. To evaluate the behavior of unexperienced users, eighteen subjects had to model an object.

After an introduction of ten minutes one group (10 persons) had thirty minutes of time to model the head of a sunflower, the other group was instructed to model a complete tulip. In this case the modelling time was 45 minutes. The first group tested mostly the iteration components while the second group was confronted mainly with geometric modelling problems. During modelling the subjects were allowed to ask questions to the supervisor. Both groups where given pictures of real sunflowers and tulips. Some of the results are shown in Figure 13.

The subjects were students of computer science, most of whom had never worked with an interactive geometry modeller before. After the test the subjects were asked some questions about the modelling process. Table 5 shows the results (Results are displayed with mean values and 90% confidential interval assuming Gaussian distribution).

	1	2	3	4	5
Intuitivity of modelling process			⊢○⊣		
Intuitivity of components				⊢○⊣	
Intuitivity of parameters			⊢○⊣		
Clarity of components				⊢○⊣	
Clarity of parameters			⊢○⊣		
Clarity of structure tree				⊢○⊣	
Editing of structure tree				⊢○⊣	
Editing of parameters			⊢○⊣		
Predictability of parameter changes			⊢○⊣		
Goal-oriented modelling				⊢○⊣	

Table 5: *Rating results of unexperienced users. (1=bad rsp. low, 5=excellent rsp. high).*

The subjects reported components and structure graph to be intuitive, clear and easy to edit. None of them had problems in connecting and arranging of components within the structure graph.
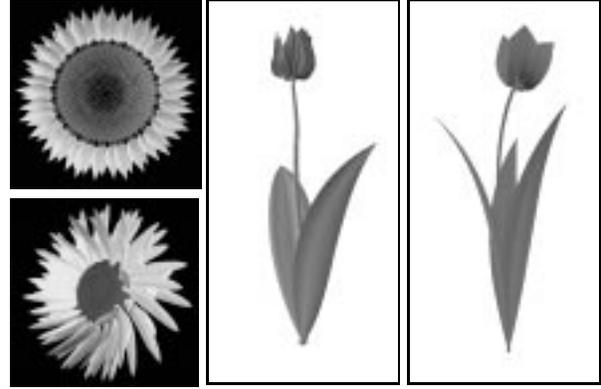
The huge parameter space caused problems as soon

Figure 13: *Results of unexperienced users. One group had to model the head of a sunflower, the other was instructed to create a tulip.*

as the subjects had to familiarize with it. The influence of parameters like different rotations where reported to be difficult to understand. For example it was sometimes not clear to the subjects whether they should rotate the whole iteration component or subsequently leave components relatively to it. The problems with understanding some of the parameters were balanced mostly by the direct feedback of the system which enables the user to try out unknown parameters efficiently.

5.1. Experienced Users

Experienced users who are familiar with the parameter set are able to model even complex objects in a short time. Table 5.1 shows the modelling times of a trained user for some of the models that were presented in this paper.

Figure	Time in h	Annotation
1	0.5	
9(a)	1.75	
9(b)	2.5	
9(c)	1.5	field definition 0.25h
10(a)	1.75	

Table 6: *Modelling times of a trained user for some of the models.*

Structure graphs can also be recycled. This makes modelling more efficient. For example, the structure graph of a typical pine can easily be used for creating another pine. By this a variety of trees of the same species can be modelled in short time. Typically, tropisms and FFD components are used to quickly change the shapes of trees.

Problems that were reported by experienced users concern mostly the influence and combination of several

tree parameters. Though complex trees can be modelled (as shown in Figure 9 and 14) the overall creation process is so far too complicated.

6. Conclusions and future work

We presented a modelling method and graphical user interface for generating natural branching objects such as plants, bushes and trees. A set of components describing geometry as well as structure is combined to form a structure graph. The user is able to model the geometry by standard interaction techniques like editing splines or using free form deformations.

Structural modelling is done by the combination of components and by specialized components which multiply others algorithmically. Global and partial constraints are integrated on the basis of tropisms and pruning operations. This can be used to elaborate specific shapes of plants and allows plants to interact with an environment.

The system was tested by experienced and novice users. Some work needs to be done to clarify and change parameters which are yet not intuitive. Future work will also focus on animating models, which is now partially integrated on the basis of key-framing techniques[1]. Level-of-detail mechanisms have to be developed to reduce the enormous geometrical complexity especially of the tree models.

The software can be obtained as a shareware[17].

7. Acknowledgements

The authors like to thank Alfred Schmitt of the University of Karlsruhe and Jeffrey Shaw of the Center for Media Technology Karlsruhe for their support in doing this work. Many thanks to Thomas Strothotte of the University of Magdeburg for giving several useful hints in improving the article and also many thanks to Sylvia Zabel for pointing out numerous spelling errors.

References

1. B. Lintermann and O. Deussen, "Interactive modelling and animation of natural branching structures", in *Computer Animation and Simulation 96* (R. Boulic and G. Hegron, eds.), pp. 139–151, Springer-Verlag, Berlin, (1996).

2. O. Deussen and B. Lintermann, "A modelling method and user interface for creating plants", in *Proc. Graphics Interface '97*, Morgan Kaufmann Publishers, (May 1997).

3. A. Lindenmayer, "Mathematical models for cellular interactions in development", *Journal of Theoretical Biology*, **I&II**, pp. 280–315 (1968).

4. P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. New York: Springer-Verlag, (1990).

5. R. Měch and P. Prusinkiewicz, "Visual models of plants interacting with their environment", in *Computer Graphics (SIGGRAPH '96 Proceedings)*, pp. 397–410, (Aug. 1996).

6. P. Prusinkiewicz, M. S. Hammel, and E. Mjolsness, "Animation of plant development", pp. 351–360, (Aug. 1993).

7. P. de Reffye, C. Edelin, J. Francon, M. Jaeger, and C. Puech, "Plant models faithful to botanical structure and development", in *Computer Graphics (SIGGRAPH '88 Proceedings)*, vol. 22, pp. 151–158, (Aug. 1988).

8. M. Holton, "Strands, gravity and botanical tree imagery", *Computer Graphics Forum*, **13**(1), pp. 57–67 (1994).

9. J. Weber and J. Penn, "Creation and rendering of realistic trees", in *Computer Graphics (SIGGRAPH '95 Proceedings)*, pp. 119–128, (Aug. 1995).

10. S. Todd and W. Latham, *Evolutionary Art and Computers.* London: Academic Press, (1992).

11. CIRAD/GERDAT, "Amap presentation." http://www.cirad.fr/amap/amap.html.

12. P. E. Oppenheimer, "Real time design and animation of fractal plants and trees", in *Computer Graphics (SIGGRAPH '86 Proceedings)*, vol. 20, pp. 55–64, (Aug. 1986).

13. Onyx Computing Inc., "Onyx tree professional." http://www. onyxtree.com.

14. PHIGS+ Committee, "Phigs+ functional description, revision 3.0", *Computer Graphics*, **22**(3), pp. 125–218 (1988).

15. P. S. Strauss and R. Carey, "An object-oriented 3d graphics toolkit", in *Computer Graphics (SIGGRAPH '92 Proceedings)*, pp. 341–347, (Aug. 1992).

16. N. Green, "Voxel space automata: Modelling with stochastic growth processes in voxel space", *Computer Graphics*, **23**(3), pp. 175–184 (1989).

17. Greenworks, "Home page of the xfrog modelling software." http://www.greenworks.de.

Appendix A: Some More Examples
To demonstrate the modelling capability of our approach, some more results are shown. Figure 14 shows a sample picture out of an animation that was created for the media artist Bill Viola. The tree is rendered in a special way to simulate the painting style of famous old Dutch painters.

Figure 14: *Tree shaded in the style of famous old Dutch painters.*

In Figure 15 a fruit fly is shown to demonstrate that even non-vegetable objects can be created. In Figure 16(b) the corresponding structure graph is shown.

Figure 15: *A non-vegetable object: the fruit fly.*

Figure 17 shows a pineapple where the segments of

the fruit were modelled as special 3-D surfaces and afterwards a texture was mapped upon them. The head of the pineapple was modelled similar to the agave shown in Figure 8.
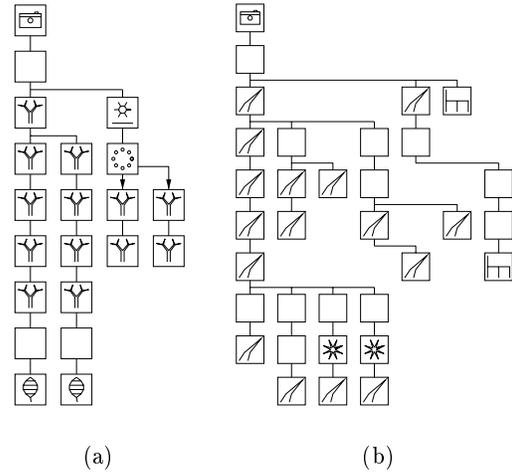

(a) (b)

Figure 16: *Structure graphs for the models of Figure 14 and Figure 15.*

Figure 17: *3-D modelling of the pineapples segments as well as applying textures can be used to enhance realism.*