3D MODELING OF TREES FROM FREEHAND SKETCH

手書きスケッチに基づく樹木の３次元モデリング

by

Makoto Okabe

岡部誠

A Senior Thesis

卒業論文

Submitted to

the Department of Information Science

on February 12, 2003

in partial fulfillment of the requirements

for the Degree of Bachelor of Science

Thesis Supervisor: Takeo Igarashi 五十嵐 健夫

Title: Assistant Professor of Information Science

Abstract

We present an interface for quickly and easily modeling 3D trees from freehand sketch. The system generates a 3D geometry from 2D sketch on the basis of our observation that trees spread their branches uniformly. Our system also supports several interfaces for editing the 3D tree by drawing additional strokes. These editing operations include adding a branch, cutting a branch and erasing a branch. Besides these interfaces, we propose a method for modeling a 3D tree by selecting branches predicted and displayed. Our system predicts and displays branches that will grow on the existent 3D tree model produced from freehand sketch and gestures. And then a user can continue to model the 3D tree by selecting the predicted branches with gestures. This interface makes it possible to model more complex and natural-looking trees.

論文要旨

手書きスケッチから迅速かつ簡単に３次元の樹木をモデリングするためのインターフェースを提案する。本手法では、一様な枝の分布を行うという樹木の性質に基づいて、ユーザの描いた樹木のスケッチを３次元的に解釈し、３次元オブジェクトの生成を行う。生成された樹木の３次元オブジェクトに対しては、スケッチとジェスチャ操作により、枝の追加、枝刈り、枝の消去、などの形状操作を行うことができる。またこれらに加えて、予測・例示される枝を選択することによってモデリングするためのインターフェースを提案する。まず、スケッチとジェスチャ操作により既に生成されている樹木モデルに対し、その樹木モデルから更に生えるであろうと予測される枝をシステムが予測・例示する。ユーザは、例示された枝をジェスチャ操作により選択し、樹木のモデリングを行う。このインターフェースを用いることで、より複雑かつ自然な樹木をモデリングできる。

Acknowledgements

I would like to say thank you to my thesis adviser Dr. Takeo Igarashi for his support in doing this work. He gave me many hints improving the article and the prototype system. And I like to thank to everybody of the Igarashi laboratory. They used my prototype system to model sample 3D trees, and sometimes encouraged me.

Contents

Chapter1

Introduction

The design of natural objects such as trees on a computer is very challenging. There are two reasons. One is the enormous structural complexity of a plant, and the other is the difficulty of 3D modeling.

A typical approach for 3D modeling of trees is to use rule-base methods. The most famous rule-base method is L-systems, introduced as a formalism for plant modeling and simulating by Lindenmayer. Modeling of plants using L-systems has reached a relatively advanced state, which is manifested by models ranging from algae to herbaceous plants and trees [1]. There are other specialized methods in reducing the structural problem and concentrating on the generation of special objects such as only trees. Weber and Penn focused on the goal of generating a visually favorable geometry [4].

These rule-based methods made it possible for expert users to design 3D models of plants and to simulate various behavior of plants, but these are not easy to use for novice users or hobby users who want to try modeling 3D trees. There are two reasons. One is that when we design 3D models of trees, we have to adjust parameters using trial and error technique or use an interactive arrangement method [3]. The other is the problem of the user interface of the system that supports only textual input.

There are systems that have GUI for editing a graph that represents a structure of a plant [5, 6, 7, 8]. This system is easy to operate a graph and change parameters of nodes

of the graph, and so users can design models in short time. However objects operated by users are graphs and parameters, still abstract.

In this thesis, our emphasis is on a viewpoint of intuitiveness and user interface. We propose a method for 3D modeling of trees from freehand sketch. All an user has to do is only drawing some strokes, gesturing and selecting some objects including buttons. And the user doesn't have to see abstract character sequences or sets of parameters.

We implemented a pen-based system with our proposed method. This system generates 3D geometry from 2D sketch on the basis of our observation how trees spread their branches. Our system also supports several interfaces for editing the 3D tree. These interfaces implemented for adding a branch, cutting a branch, and erasing a branch. Besides these interfaces, we propose a method for producing more complex and natural-looking trees. Simple production rules are derived from the existent 3D tree model generated from a freehand sketch and gestures. The rules can be used to predict and to display branches that will grow on existing 3D tree model. By selecting predicted and displayed branches, a user can design a more complex and more natural-looking 3D tree. Modeling on our system, a novice user can design 3D trees easily and intuitively.

Our method doesn't assume that the input sketch represents the whole tree and is used to reconstruct the complete 3D tree model at once. The method that we propose in this thesis is more interactive. We want to research the method for reconstructing the complete 3D tree model from freehand sketch at once in the future.

The remainder of this thesis organized as follows: Previous Work section provides relevant information on 3D modeling of trees, 3D modeling from freehand sketch, and beautification and prediction technique. User Interface for Modeling Trees section explains about our pen-based system implemented with our proposed method. Data Structure and Algorithm section explains about algorithms and data structures used on our system. Result And Estimation section presents samples of 3D trees designed on our system and comments expressed by some users of our system. Conclusion section concludes with a discussion of contributions. Limitations And Future Work section

describes limitations of our method and possible directions for future work.

Chapter 2

Previous Work

2.1. 3D Modeling of Trees

In 1968, biologically motivated, Aristed Lindenmayer proposed the formalism of L-systems as general framework for plant modeling, and plant modeling emerged as various area of interdisciplinary research, such as biology, plant science, mathematical, and computer science [1]. An L-system is a parametric rewriting system operating on branching structures represented as bracketed strings of modules. In the context of L-systems, the term module denotes any discrete constructional unit that is repeated as the plant develops, for example a branch or a flower. Simulation begins with an initial string called the axiom, and proceeds in a sequence of discrete derivation steps. In each step, rewriting rules or productions replace all modules in the predecessor string by successor modules. The goal of modeling at the modular level is to describe the whole structure of a plant.

Prusinkiewicz extended L-systems in a manner suitable for simulating the wide range of interactions between a developing plant and its environment, and proposed OPEN L-SYSTEMS [2]. They implemented models that capture collisions between branches, the propagation of clonal plants, the development of roots in soil, and the development of tree crowns competing for light. Today the resulting models are used as a research tool in biology and lead to increasingly convincing visualizations.

While researches described above concentrated on finding efficient descriptions that fit into not only principles of trees but also botanical principles, the emphasis of Weber and Penn is on the overall geometrical structure of the tree and not a strict adherence to botanical principles [4]. In their method, a set of textual parameters describes the geometry for each branching level of the tree.

These rule-based systems make it possible for expert users to model plants of various kinds and simulate them. However, for a novice user, the interface of each system is not easy to use.

While rule-based researches of plant modeling listed above are premised on textual parameters as user interface, Oliver Deussen and Bernd Lintermann presented a system developed for the purpose of combining the power of a rule-based approach with the intuitiveness of generic tree methods [5, 6, 7, 8]. On this system, a graph describes the structure of the plant. Nodes of the graph are components representing parts of the plant, and edges denote creation dependencies. Global and partial constraint techniques are integrated on the basis of tropisms and allow the modeling of specific shapes. Using this system, a user is able to model the geometry by standard interaction techniques like editing splines or using free form deformations. This system is named Xfrog and available as shareware at URL: http://www.greenworks.de.

This system makes it possible for a non-expert user to model plants of various kinds, but an user has to operate abstract objects like a graph, nodes and parameters.

Alias|Wavefront K.K have Maya, which is a software for 3D modeling and animation. Paint Effects is one of the many plug-ins of Maya. How to use Paint Effects is that a user selects a brush and draws strokes. Images of the selected brush are putted along the strokes in 2D mode, and 3D objects of the selected brush are putted along the strokes in 3D mode. Maya have various brushes for Paint Effects, for example animals, flowers, hairs, plants, trees, and weathers. By this plug-ins, a user can design the hair of

an animal model or plants grown on a rock model. Besides the modeling capability, this plug-ins supports interfaces for animation. A user can easily make animations, for example hair or plants blown on a wind.

Paint Effects of Maya is very useful to model and simulate plants already implemented as brushes, but it is another problem to design an original plant model.

Our method proposed in this thesis is based on these previous researches listed above and aims at modeling 3D tree by drawing sketch and gesturing.

2.2. 3D Modeling from Freehand Sketch

When a 3D tree is modeled on our system, the input by an user is always on the 2D screen. And so one of the most important tasks of our system is to decide the depths of the user's 2D input.

Previous researches proposed several methods that a computer could make 3D models by interpreting inputted 2D strokes as parts of an assumed 3D model.

Lipson proposed an optimization-based algorithm for reconstructing a 3D model from a single, inaccurate, 2D edge-vertex graph [10]. The graph is obtained from an inaccurate freehand sketch of a 3D wire-frame object. Since the inputted sketch, assumed to represent a projection of a general object as seen from some arbitrary point of view, lacks depth information, the reconstruction process is non-deterministic and its main task is to determine the depths of the vertices. At first, Lipson proposed the identification method of faces in a 2D projection of a 3D wire-frame model [11]. Then using the surface information derived by the former method, he proposed several constraints used to determine the depths of the vertices. The constraints are for example face planarity, line parallelism and line orthogonality. When a simple sketch of an industrial design is inputted, the system implemented with this method can reconstruct

3D model for reasonable time.

Igarashi demonstrated the 2D-to-3D idea with Teddy system [9]. When a closed 2D stroke is inputted, Teddy makes a seemly 3D polygon model by calculating the depths of the surfaces with law learned by experience. Besides this creation function, Teddy supports several interfaces for editing the model by drawing additional strokes. These editing operations include extrusion, digging a cavity, turning the closed stroke into a surface drawing, cutting operation, smoothing operation and morphing operation. Using this system, a user can design models of free-form surfaces like stuffed animals.

2.3. Beautification and Prediction Technique

Igarashi proposed a technique for rapid geometric design by interactive beautification and by predictive drawing [12]. The prototype system, named Pegasus, is implemented with this technique. Interactive beautification is to beautify a freehand stroke on the basis of geometric constraints, for example parallelism or perpendicularity. And predictive drawing is to predict and display the multiple drawing candidates associated from the existent figure that the user has already drawn. If the drawing candidate suggested by this system matches the user's intension, the user can design the geometric figure by selecting the drawing. The user of this system can design geometric figures in a shorter time than users of other systems.

Igarashi extended his Pegasus system to design 3D model [13]. Like Pegasus, this system, named Chateau, suggests operations that the user is likely to do next. Pegasus displays its suggestions on the drawing figure, but this system displays small images of the suggestions in rectangle areas for keeping the modeling area clear. If the operation suggested by this system matches the user's intension, the user can easily continue to design the 3D model by selecting the suggested image.

From a viewpoint of modeling 3D tree, we propose the prediction method on the basis of these previous researches. Our method predicts branches that will grow on the existent 3D tree model.

Chapter 3

User Interface for Modeling Trees

This section describes the modeling operations of the system from the user's point of view; details of the algorithms are left to the next section.

3.1. Overview

Figure 3.1 introduces the first stage of 3D modeling of trees on our system. The user begins by drawing a pair of strokes on a blank canvas (a). These two strokes are supposed to be outlines of a trunk (b). After drawing a trunk, the user can add branches on it (c). When the user rotates the sketch, the branches spread in 3D space, and 3D polygonal object of the tree is produced (d).

Figure 3.2 shows the process of adding a branch on existent 3D tree model. A user draws outlines of a branch at an adequate viewpoint (a), and then a 3D branch object is created (b).

Figure 3.3 shows the process of cutting and erasing a branch. A user draws a single stroke crossing an existent branch by one point (a), and then the branch is cut (b). A single stroke that intersects a branch by multiple points is drawn (c), and then the branch is erased (d).

Figure 3.4, 3.5, 3.6 and 3.8 show the modeling process of selecting branches predicted by our system around a trunk or a branch other then a trunk. When a user

double-clicks a branch, predicted branches are generated around the branch, and a user can continue to model a 3D tree by selecting them. In Figure 3.4, after a simple model is modeled like (a), a user can double-click a trunk, and then predicted branches are generated around it (b). The user can select branches needed for the model with a single stroke (c), and selected branches are added to the tree structure (d). Figure 3.5 shows that a user can select a predicted branch with a single-click and Figure 3.6 shows that when a user double-clicks the trunk again without selecting any predicted branches, our system predicts a new branch generation based on the next shape pattern. In Figure 3.8, a user double-clicks a child branch in (a), and then branches are predicted around the branch (b). While the user can select the branches (c), the user can double-clicks the child branch again (d), and then the state of the clicked branch is propagated over the other child branches.

3.2. First Stage of 3D Modeling of Trees

Figure 3.1 introduces the first stage of the model construction process of the system. A branch is made by two strokes drawn with a mouse or a pen on a tablet. At first, the canvas of our system is blank (a). The user begins the design of a 3D tree model with drawing a pair of strokes that represents outlines of a branch (b). After the first branch, as a trunk, is drawn on the canvas, the user can add child branches on it (c). Grandchild branches can be added on child branches likewise. The first branch must be a trunk, so the user cannot begin drawing a child branch.

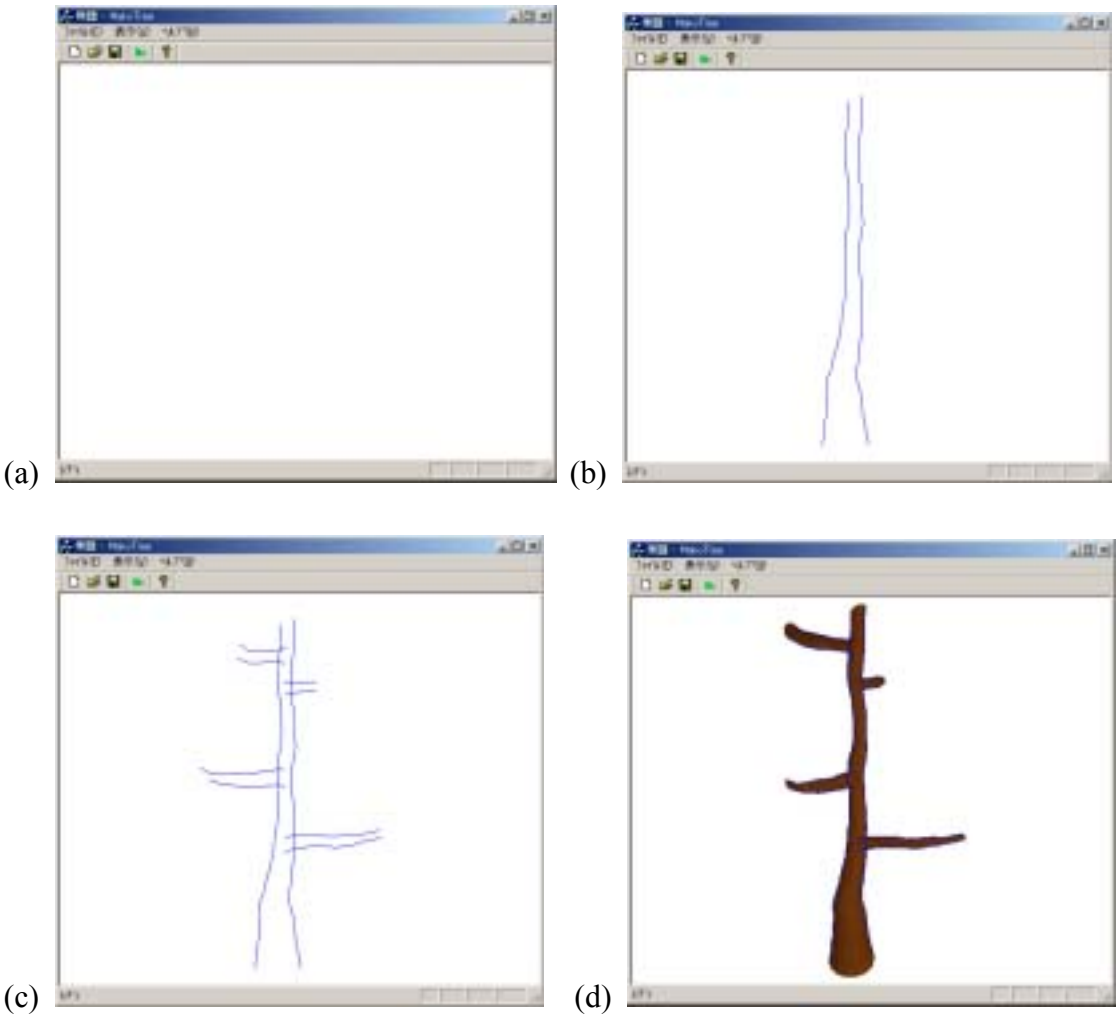


Figure 3.1: (a) initial state, (b) drawing two strokes of a trunk, (c) addition of four branches, (d) result of the 3D tree model

Basically, the design process is supposed to advance to the deep level of the tree structure. The reason why we implemented the fundamental part of our system like this is based on the observations of people sketching trees simply. When we sketch a tree not in detail but simply, we tend to begin sketching a trunk. And after adding some branches on the trunk, we tend to draw vague strokes representing leaves.

Each additional branch must be a child of a branch existing on canvas already. So

even if the additional branch is drawn apart from the other branches, after searching the nearest branch among them, the additional branch is supposed to move on it.

When you finish drawing strokes of branches and rotate the sketch, the sketch is converted to the 3D model (d). Child branches are spread in 3D space by taking into consideration a simple rule that child branches of a tree spread uniformly. Its detail explanation is left to the next section.

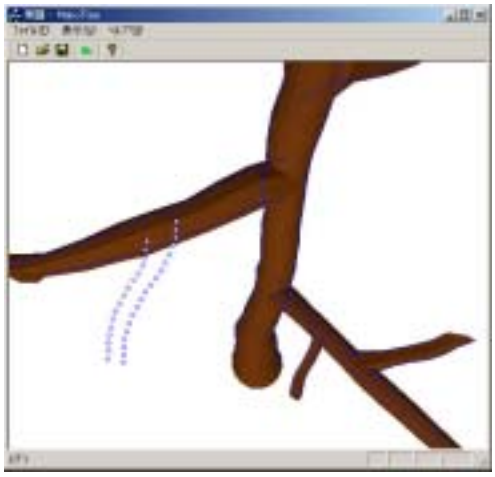
(a)  (b) 

Figure 3.2: (a) strokes of an additional branch, (b) strokes converted to 3D model immediately

Figure 3.2 shows that the user can add branches on the 3D tree made from the first sketch. After adjusting the viewpoint, the user can draw a pair of strokes in the same way as shown above (a). And then the strokes are moved so that the branch that they represent grows from the parent branch, and a new 3D branch object is made from the strokes (b). The new branch is supposed to be parallel to the screen of the canvas.

3.3. Cutting and Erasing Branches

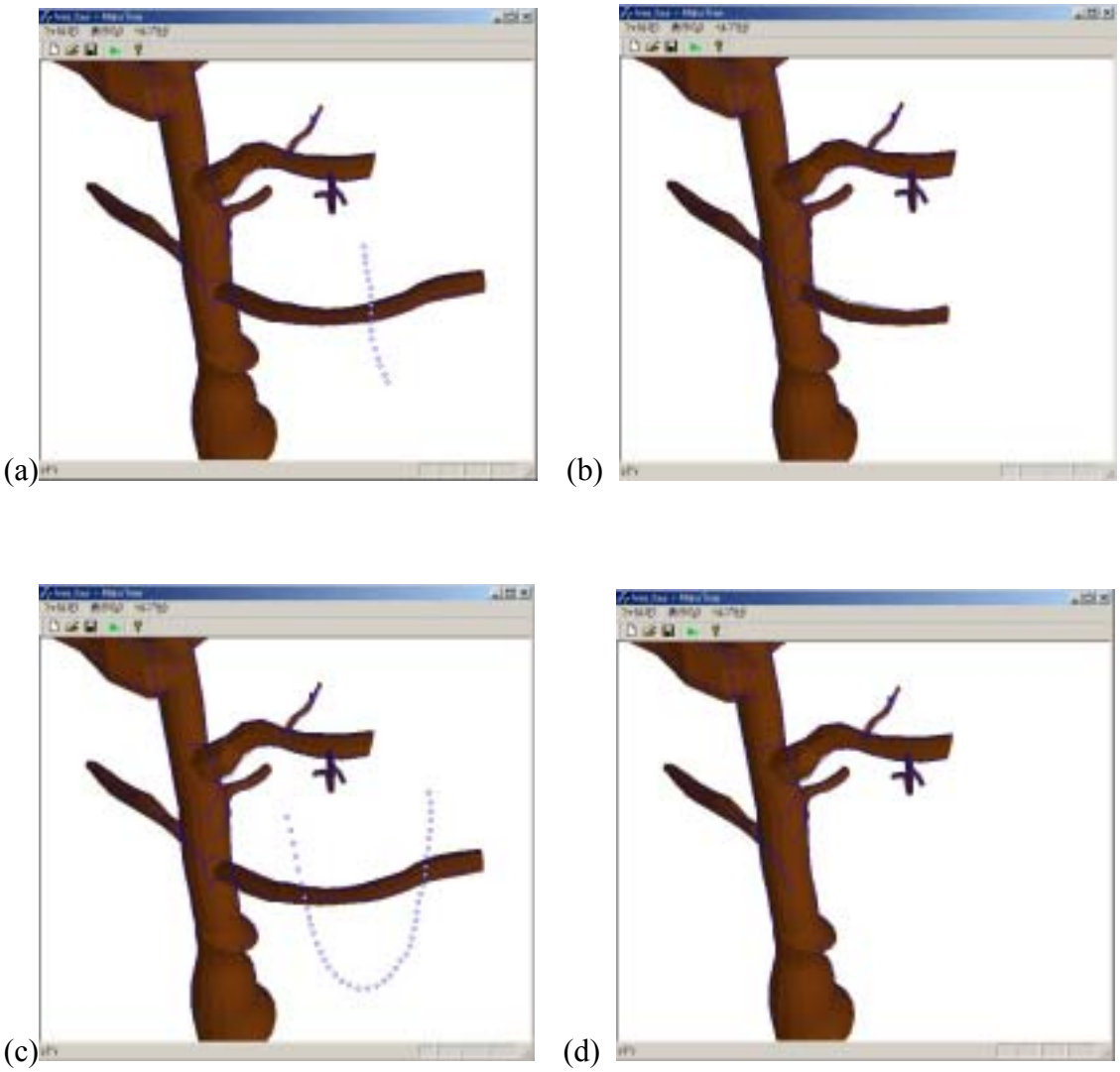


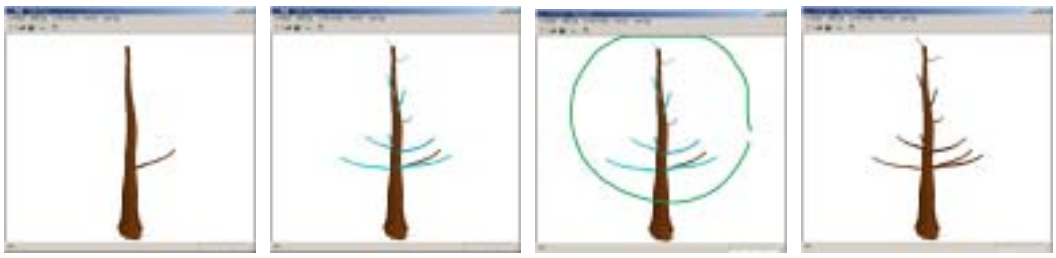
Figure 3.3: (a) a stroke for cutting, (b) result of cutting, (c) a stroke for erasing, (d) result of erasing

A user can cut or erase the existing branch, and Figure 3.3 introduces the cutting and erasing process. When a stroke intersecting the objective branch by one point is drawn (a), the branch is cut and the tip of the branch is deleted (b). When a stroke

crossing the objective branch by two points or more is draw (c), the branch is erased (d).

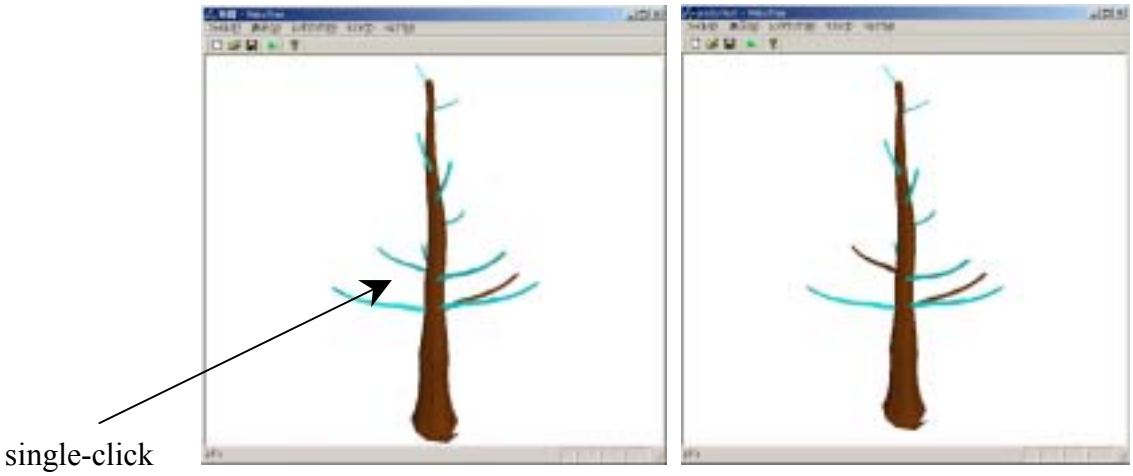
3.4. Selecting Predicted Branches

We propose a modeling method where the system predicts and displays branches and a user can design the 3D tree by selecting the branches suggested by the system. On our system, the double click event is used for a user to indicate the branch on which the prediction operates. Figure 3.4, 3.5 and 3.6 show the modeling process by selecting predicted branches around the trunk.



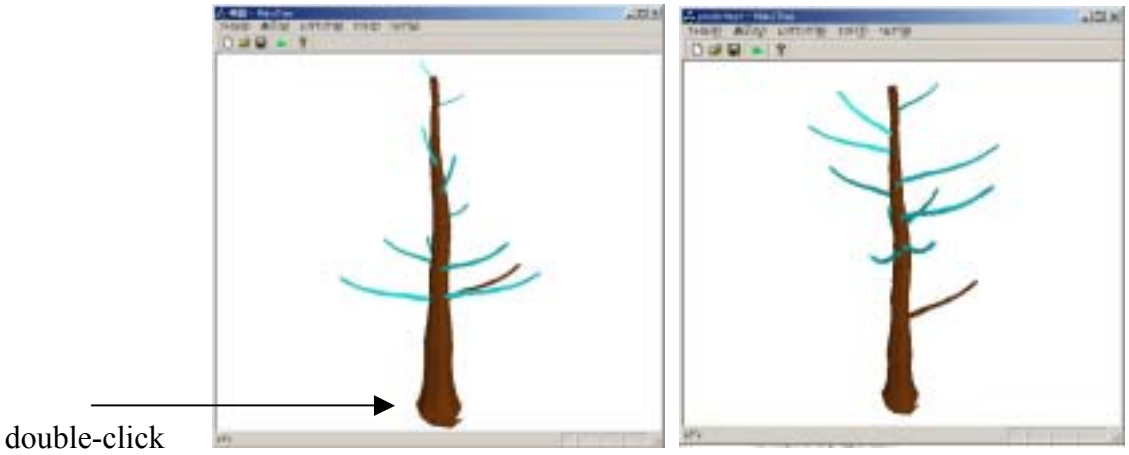
(a) (b) (c) (d)

Figure 3.4: (a) initial state, (b) predicted branches around the trunk, (c) selecting all branches, (d) completion model



(a) (b)

Figure 3.5: (a) predicted branches around the trunk (same as Figure 3.4 (b)), (b) selecting a single branch



(a) (b)

Figure 3.6: (a) predicted branches around the trunk (same as Figure 3.4(b)), (b) prediction of the next pattern

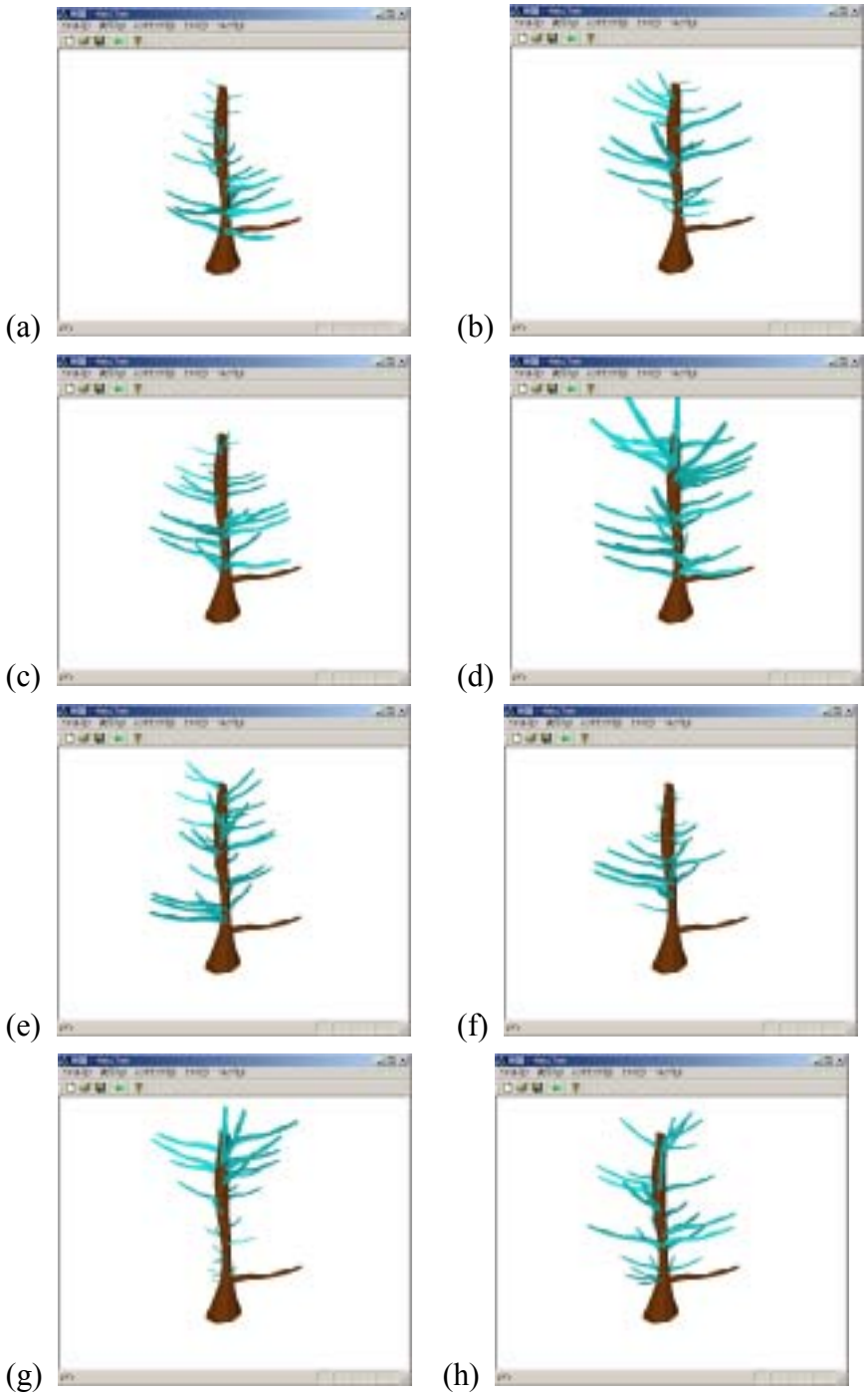


Figure 3.7: There are eight patterns of branch generation. (a) conical, (b) spherical, (c) hemispherical, (d) cylindrical, (e) tapered cylindrical, (f) flame, (g) inverse conical, (h) tend flame

Figure 3.4 (a) shows a simple 3D tree model, which consists of a trunk, a single child branch. When a branch is selected, by double-clicking on our system, the prediction process is launched. The trunk is just double-clicked in Figure 3.4 (b), and then the result of branch prediction is immediately displayed with light-blue branches. The prediction on the trunk has eight patterns [4]. The patterns represent tree's shapes, which are "conical" (a), "spherical" (b), "hemispherical" (c), "cylindrical" (d), "tapered cylindrical" (e), "flame" (f), "inverse conical" (g) and "tend flame" (h), and Figure 3.7 shows these patterns. When the trunk is double-clicked without selecting any branch, the prediction pattern changes in the order described above (Figure 3.6), but which pattern is being used is not shown to the user. And two methods are provided for selecting branches. First method is to single-click an objective branch, and second method is to enclose the objective branches with a single stroke. Figure 3.4 (c) and (d) show the first method, and Figure 3.5 shows the second method. Selected branches change its color into brown and they are added to the tree structure.

In the case of branches other than the trunk, the operation described above for changing the prediction pattern can be used to generate many similar branches, and so there is no pattern used in the prediction process. The modeling process by this operation is shown in Figure 3.8.

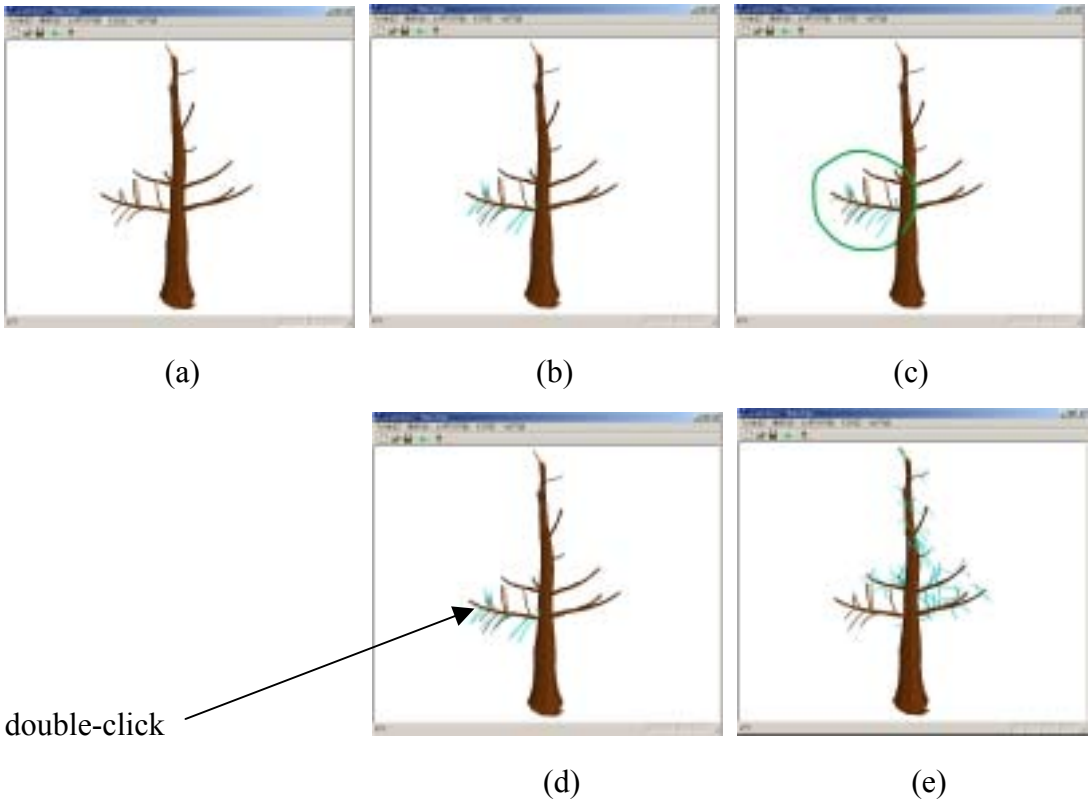


Figure 3.8: (a) initial state, (b) predicted branches around the left-bottom branch, (c) selecting all predicted branches, (d) same as (b), (e) copying the left-bottom branch's state over the all child branches

Figure 3.8 (a) shows a tree model that has a trunk, several child branches and several grandchild branches. Figure 3.8 (b) shows that first double-click event on the left-bottom branch launches the prediction result on the branch. Figure 3.8 (c) shows the selecting process by enclosing the predicted branches, that is the same operation as described in the case of the trunk. If the user double-clicks the left-bottom branch again at the state of Figure 3.8 (d), the left-bottom branch's state is copied over all child branches (Figure 3.8 (e)). A user can model a more complex tree efficiently by selecting the predicted and propagated branches.

Chapter 4

Data Structure and Algorithm

4.1. Data Structure of 3D Tree Model

On this system, the data structure memorizing the 3D tree model is literally the tree structure. Each node represents the branch, for example the root node represents the trunk. Each node has information of its representing branch. The information includes the outlines, the medial axis, the 3D polygonal object, the normalized position from 0.0 to 1.0 on the parent branch, and so on.

4.2. Make 3D Branch by Extrusion

The method of making 3D branch consisting of polygons is the extrusion algorithm [9]. The fundamental idea of this method is to put copies of a 2D figure, for example a regular polygon, along a path and to sew them with polygonal meshes for creating a pipe-like 3D object. See Figure 4.1.

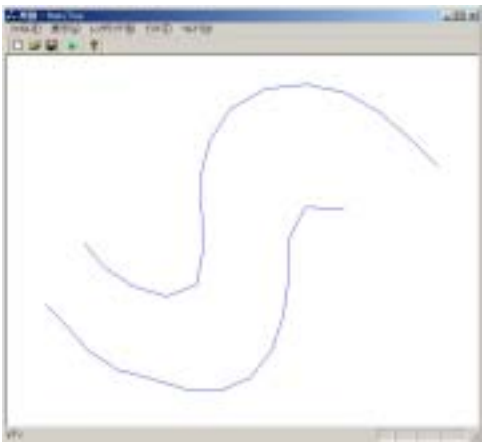
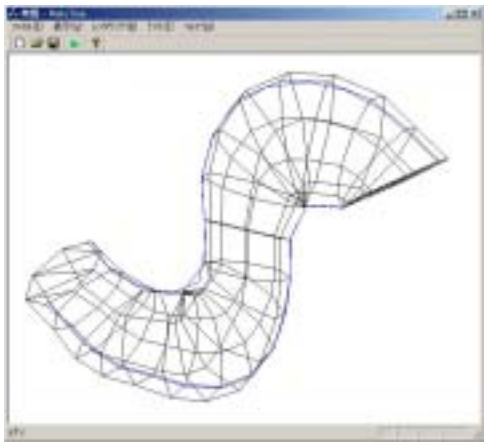
(a)  (b) 

Figure 4.1: (a) outlines of a branch, which an user draws, (b) result of making a 3D branch object by the extrusion algorithm

On our system, regular polygons are sewed with polygonal meshes for creating a 3D branch. Two strokes representing the outlines of a branch are used to calculate the central positions, the sizes and the direction vectors of the regular polygons. The set of the central positions is the extrusion path.

4.3. Spread 3D Branches in 3D Space

When a user draws two strokes of a branch, a node for the branch information is made and added to the tree structure. When the user finishes drawing the sketch and rotates the sketch, each branch is spread in 3D space, and a 3D polygonal object is made for each branch. During this process, the sketch from the first viewpoint is not changed. Figure 4.2 shows how branches spread in 3D space.

(a)  (b) 

Figure 4.2: (a) the initial sketch, (b) seeing the spread branches from the top of the trunk

Each branch exists on either left-hand side or right-hand side of its parent branch. The branches are supposed to spread so that each branch is at the equal angle with its neighbor branch on each side. And a shorter branch is supposed to spread to the more acute angle direction from the sight vector and to spread on back side, in order to balance the result 3D tree. For example, in Figure 4.2, five branches are drawn around the trunk. Two branches are on left-hand side and three are on right-hand side. So each branch on left-hand side is at 180/3(=60) degrees, and it on right-hand side is at 180/4(=45) degrees.

We consider that this process can be improved. Seeing Figure 4.2, the top-right branch spread in 3D space becomes too long. According to our observations of people sketching trees simply, this is because a user tends not to draw branches that grow forward or backward, which is considered as the key to improve this process.

4.4. Cut and Erase 3D Branch

On this system, the cutting operation is not the boolean operation in a strict

meaning. Actually the cutting operation cuts the outlines of the branch, calculates the medial axis again, and reconstructs the 3D polygonal object of the branch. This is because this system always needs the outlines and the medial axis of a branch as described above, and so the outlines are the most important information for the branch.

The erasing operation is to search the node that has the information of the objective branch and to delete the node from the tree data.

4.5. Automatic Branch Generation

Child branches that are likely to spawn on a branch are supposed to be predicted and displayed on this system. We developed this prediction algorithm on the basis of [4]. The algorithm for predicting a child branch spawning on its parent branch is as follows:

First, this algorithm decides the position of the child branch on its parent branch. This position is calculated so that the child branch spawns between the most separated pair of branches among the existent branches on the parent branch. In the case that the parent branch is the trunk of the tree, the position of the child branch is above the position of the lowest branch among the existent branches on the trunk.

After deciding the position of the child branch, the length of the child branch is calculated. See “4.3 Stem Children” section of [4] for details. Its calculation differs in the case of the trunk and in the case of other branches. In the case of the trunk, the lengths of children are dependent on the shape of the tree that the user is modeling. The “ShapeRatio” function is defined for calculating the length, and the function has nine types in [4]. We use eight types of them, except for “envelope”, which are “conical”, “spherical”, “hemispherical”, “cylindrical”, “tapered cylindrical”, “flame”, “inverse conical” and “tend flame” (Figure 3.7).

The next process is to calculate the vector, called “child vector”, along which the child branch spawns. Likewise, the vector along which the parent branch spawns is called “parent branch”. This process consists of two steps: Step one is to calculate the

perpendicular ingredient of the child vector to the parent vector. This ingredient is called “rotation vector”. Step 2 is to calculate the angle, called “down angle”, at which the child vector is to the parent vector. The calculation of the rotation vector differs in the case of the trunk and in the case of the other branches. In the case of the trunk, the rotation vector is calculated so that the children spread uniformly. In the case of other branches, this vector tends not to turn to bottom. We consider the vector along which the trunk stands, called “trunk vector”, as the vector turning to the sun, and calculate the rotation vector so that it is at 140 degrees with the trunk vector by 80% of probability. The down angle is calculated as a random value in the range of the minimum down angle to the maximum among the existent branches on the parent branch.

At the last, the new branch is made on the basis of the information described above and added to the tree data. Besides the information, outlines defining the form of a branch are needed to make the new branch. In order to decide the form of the new branch, a branch is chosen randomly from the existent branches on the parent branch, and the outlines of the chosen branch are copied as the outlines of the new branch.

Chapter 5

Results

Our prototype system is developed mainly for the purpose of modeling branches of 3D trees, but it also provides a simple interface for putting polygonal leaves on 3D tree models. A leaf is represented with a square polygon of a user-selected color. And leaves are supposed to be attached to all branches that belong to the leaf nodes of the tree structure. Figure 5.1 shows sample tree models designed on our system by the author, and Figure 5.2 shows sample tree models designed on our system by test users.

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

Figure 5.1:

(a), (b) a Japan cedar with 687 branches consisting of 31548 polygons

(c), (d) a cherry blossom with 945 branches consisting of 49634 polygons

(e), (f) a Ulmus japonica with 5643 branches consisting of 77529 polygons

(g), (h) an unnamed tree with 2237 branches consisting of 38997 polygons

(i), (j) a Ficus benjamina with 1418 branches consisting of 34533 polygons

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

Figure 5.2:

(a), (b) a persimmon tree with 368 branches consisting of 8928 polygons

(c), (d) an unnamed tree with 658 branches consisting of 13098 polygons

(e), (f) a pine-like tree with 68 branches consisting of 2877 polygons

(g), (h) an unnamed tree with 7849 branches consisting of 118549 polygons

(i) a cactus-like tree with 1826 branches consisting of 40828 polygons

We ask some users to use our system and express some comments:

- When there are many branches, the processing becomes heavy.
- When using a pen on a tablet, double-click event is not easy to use.
- A fine branch should be drawn with a single stroke.
- The global shape of a tree should be defined or deformed easily.
- Branches that are the latest generation can be leaves, so the system should have an interface for drawing a leaf and propagating the leaf over the whole tree.
- If prediction and propagation are visualized on real time as if the tree model was growing up, it will be joyful and effective for modeling process.

Chapter 6

Conclusion

We have introduced the modeling and editing method of 3D trees from freehand sketch. A user can model a tree with more complex structure by using prediction and propagation method, and the prediction and propagation are based on the first model from sketch, so the completion model always has the character that the first sketch also has. A user can model a 3D tree freely and creatively on this system.

Chapter 7

Limitations and Future Work

So far, the user's sketch is used only at the first stage where a trunk and a few child branches are drawn, except for drawing additional branches. One of the reasons is that our system is not able to reconstruct a 3D tree model from a 2D sketch well. So we want to develop the method for reconstructing a 3D tree model from a more complex 2D sketch at once (Figure 7.1).

(a)  (b) 

Figure 7.1: (a) an user's sketch as the input, (b) result of reconstructing a 3D tree model from the sketch at once

We concentrated on the research about the method how to model branches efficiently by means of sketches and gestures in this thesis, so we described little about the modeling method of leaves. Although we have already attempted several methods for modeling leaves, we couldn't have found a good method. But when we attempted those methods, our approach was based on the idea of modeling a 3D tree from only a freehand sketch without other operations. So our next approach is how to model efficiently leaves of a tree over a more complex 3D tree model that is designed by the method described in this thesis.

Our approach doesn't expect the user to draw an exact sketch of a tree, and the user can design a 3D tree model freely and creatively. So some users may want to render their 3D tree model non-photorealistically. The method of non-photorealistic rendering of 3D tree model is researched in [14], and various methods aimed at various 3D models have researched. So we plan to find the adequate method to render the model created on our system and to implement it.

While a user can model various trees of the same kind by using L-systems, all our system can do is to assist the user in modeling a single tree so far. We are considering how our system can be extended so that it is able to help a user to model various trees of the same kind. We are researching about the method for deriving some rules from the 3D tree model generated on our system and for using the rules to model various trees of the same kind.

Reference

[1] Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, and Radomir Mech. “L-systems: from the theory to visual models of plants” Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences. CSIRO Publishing, 1996.(To appear)

[2] Radomir Mech and Przemyslaw Prusinkiewicz. “Visual Models of Plants Interacting with Their Environment” Proceedings of SIGGRAPH 96, pp.397-410, 1996.

[3] Joanna L. Power, A.J. Bernheim Brush, Przemyslaw Prusinkiewicz, David H. Salesin, “Interactive Arrangement of Botanical L-System Models” Proceedings of the 1999 symposium on Interactive 3D Graphics, pp.175-182, 1999.

[4] Jason Weber, Joseph Penn, “Creation and Rendering of Realistic Trees” Proceedings of SIGGRAPH 95, pp.119-128, 1995.

[5] Oliver Deussen, Bernd Lintermann, “Interactive Modelling of Branching Structures” ACM SIGGRAPH 96 Visual Proceedings, p 148, 1996.

[6] Oliver Deussen, Bernd Lintermann. “A Modelling Method and User Interface for Creating Plants” Proceedings of Graphics Interface 97, pp.189-197, 1997.

[7] Oliver Deussen, Bernd Lintermann. “Interactive Modeling of Plants” IEEE Computer Graphics and Applications, 19(1):pp.56-65, 1999.

[8] Greenworks GbR. Home page of the xfrog modeling software. http://www.greenworks.de.

[9] Takeo Igarashi, Satoshi Matsuoka, Hidehiko Tanaka, "Teddy: A Sketching Interface for 3D Freeform Design" Proceedings of SIGGRAPH 99, pp.409-416, 1999.

[10] Hod Lipson, Moshe Shpitalni, "Optimization-Based Reconstruction of a 3D Object From a Single Freehand Line Drawing” Journal of Computer Aided Design, Vol.28 No.8, pp.651-663, 1996.

[11] Hod Lipson, Moshe Shpitalni, "Identification of Faces in a 2D Line Drawing

Projection of a Wireframe Object" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.18, No.10, pp.1000-1012, 1996.

[12] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, Hidehiko Tanaka, “Interactive Beautification: A Technique for Rapid Geometric Design” Proceedings of UIST ’97, pp.105-114, 1997.

[13] Takeo Igarashi, John F. Hughes, “A Suggestive Interface for 3D Drawing” Proceedings of UIST '01, pp.173-181, 2001.

[14] Oliver Deussen, Thomas Strothotte, “Computer-Generated Pen-and-Ink Illustration of Trees” Proceeding of SIGGRAPH 2000, pp.13-18, 2000.