Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Flower solid modeling based on sketches*

Zhan DING†, Shu-chang XU, Xiu-zi YE†‡, Yin ZHANG, San-yuan ZHANG
(*State Key Lab of CAD & CG, School of Computer Science, Zhejiang University, Hangzhou 310027, China*)
†E-mail: dingzh@hotmail.com; yxz@zju.edu.cn

Received Aug. 21, 2007; revision accepted Oct. 29, 2007

Abstract: In this paper we propose a method to model flowers of solid shape. Based on (Ijiri *et al.*, 2005)'s method, we separate individual flower modeling and inflorescence modeling procedures into structure and geometry modeling. We incorporate interactive editing gestures to allow the user to edit structure parameters freely onto structure diagram. Furthermore, we use free-hand sketching techniques to allow users to create and edit 3D geometrical elements freely and easily. The final step is to automatically merge all independent 3D geometrical elements into a single waterproof mesh. Our experiments show that this solid modeling approach is promising. Using our approach, novice users can create vivid flower models easily and freely. The generated flower model is waterproof. It can have applications in visualization, animation, gaming, and toys and decorations if printed out on 3D rapid prototyping devices.

Key words: Solid modeling, Floral diagram, Inflorescence, Gesture, Constrained Delauny triangulation (CDT), Freehand sketching
doi:10.1631/jzus.A071443 **Document code:** A **CLC number:** TP391.7

INTRODUCTION

There has been a great deal of work on plant modeling and simulation. These include planet structure and shape modeling, visualization, plant development animation, plant motion animation under external force, and real-time planet realistic rendering. However, due to their complex shape and wide variation in appearance, plants remain one of the most difficult objects to model. Prusinkiewicz and Lindenmayer (1990) proposed the L-system to generate plants of complicated structures. L-system describes an object by an axiom and a set of production rules, which can be called a grammar since it describes the object structure. The grammar can derive in subsequent iterations 3D objects of a given structure. Furthermore, by using the turtle graphics, the symbolic objects can be visualized. Modeling using L-system requires users to have knowledge of formalism in L-system.

Ijiri *et al*.(2005) proposed a sketch-based interface to model flowers. It separates modeling procedure into two independent parts: structure modeling and geometry modeling. The structure editor consists of two subsystems: one for individual flowers driven by floral diagram; another for layout of multiple flowers driven by inflorescence. The geometry editor focuses on modeling of several botanical elements such as branch, pistil, receptacle, petal, calyx and stamen.

Models generated using above existing methods can be well-suited for decorating in architectural design and layout, gaming environment, or scientific study of plant growth and blooming. In this paper, we focus on generating flower's solid shapes that are as faithful to their botanical nature as possible. We take a sketch-based approach, and our initial modeling procedures are similar to those in (Ijiri *et al*., 2005). However, we emphasize here on generating solid waterproof geometrical shapes so that the local details

‡ Corresponding author
* Project supported by the Hi-Tech Research and Development Program (863) of China (Nos. 2007AA01Z311 and 2007AA04Z1A5), the Postdoctoral Science Foundation of China (No. 20070421185), and the National Research Foundation for the Doctoral Program of Higher Education of China (No. 20060335114)

of the flowers are realistic and can be directly printed onto rapid prototyping devices for creating realistic decorations or toys.

PRIOR WORK

Plant modeling

Many efforts have been devoted to the modeling of plant structures and shapes in recent years. The most fundamental work was done by Prusinkiewicz and Lindenmayer (1990). Lindenmayer system (L-system) is the basis for lots of subsequent plant modeling work. There have been many extensions of the original L-system that add flexibility and power to create more advanced and higher-level structures. These extensions include the bracketed, stochastic and context-sensitive L-system (Prusinkiewicz and Lindenmayer, 1990), the physics enhanced L-system (Noser *et al*., 2001) and the environmentally sensitive extension (Prusinkiewicz *et al*., 1994). Boudon *et al*.(2003) introduced a method to create bonsai trees more intuitively and interactively using L-systems. Ijiri *et al*.(2005) separated flower modeling procedures into individual flower modeling and inflorescence modeling. Quan *et al*.(2006) proposed a semi-automatic method for modeling plants directly from images. Okabe *et al*.(2005) presented a plant modeling system totally based on freehand sketches and additional example-based editing operations.

Sketching interface

Sketch-based modeling methods have gained popularity recently. Sketching interface provides users an easy way to create a rough object instead of creating a precise model using traditional WIMP (Window, Icon, Menu and Pointing device) interactive way. Many smart methods have been proposed for constructing 3D models from user-defined 2D sketches. These include the reconstruction of rectilinear models covered by planar faces by solving constraints (Eggli *et al*., 1997) or using optimization-based algorithms (Lipson and Shpitalni, 1996), the reconstruction of 3D curves using energy minimization (Pentland and Kuo, 1989), symmetric relationship (Tanaka *et al*., 1989) or using screen projected curve and its shadow (Cohen *et al*., 1999), and sketch- and constraint-based editing of free-form curves and surfaces (Michalik *et al*., 2002). SKETCH (Zeleznik *et al*., 1996) provided a gesture-based interface to design 3D scenes consisting of CSG-like simple primitives, and Teddy (Igarashi *et al*., 1999) allows users to design freeform models easily.

In this paper, we take a similar approach to (Ijiri *et al*., 2005) in modeling flower shapes. However, both structure diagrams are expressed using specific 3D shapes in our system. We focus on interactive sketching interfaces for designing 3D objects using 2D sketches and implicit gestures to allow users to freely and easily edit the geometrical parameters directly on floral diagrams and inflorescence diagrams using the predefined implicit gestures.

OVERVIEW OF OUR SOLID FLOWER MODELING SYSTEM

Our flower modeling system uses sketches and some predefined structural diagrams. There are three independent components in our system: structure modeling, geometric modeling, and solid generation. We use inflorescences diagrams and floral diagrams to model inflorescences structure and individual floral structure, respectively. To facilitate this process, we designed a rich set of implicit gestures so that a user can freely and rationally edit any botanical parameters directly onto these structural diagrams. Our inflorescences and floral structural diagrams are both expressed using specific 3D shape.

Geometrical modeling of a flower contains pistil, stamen, petal/calyx, receptacle and stem modeling (Fig.1). Here pistil is modeled as a revolved surface; stamen is modeled as a swept surface (spermaduct) and an inflation object (anther); petal is expressed as a B-spline surface; and stem is expressed as a swept surface similar to spermaduct. All these components are created from or edited by sketches using the similar methods proposed by Ijiri *et al*.(2005). Using dragging gestures we can drag any geometrical component to a proper position in the structural diagrams. Then those new positions are mapped onto independent components of the flower models. Geometry and structure are finally associated to each other through this mapping operation.

The final step of modeling is to generate flower solid shape. The main operation is merging, including

stem-stem merging, spermaduct-anther merging, and flower inner merging. After the merging operation, all components in the flower model are combined into a single waterproof shape.

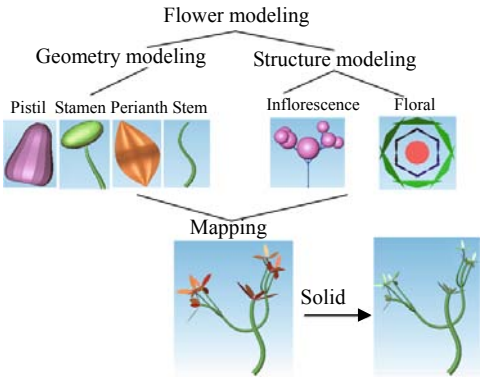


Fig.1 Overview of our solid flower modeling method

STRUCTURAL AND GEOMETRICAL MODELING OF FLOWER

Flower modeling is separated into two independent parts: inflorescences and individual floral modeling.

Inflorescences modeling

In our system, we predefine four types of inflorescences diagrams: raceme, umbel, dichasium and drepanium. The other types of inflorescences referred to in (Bell, 1991) can be easily created based on the above four patterns. The compound inflorescences can be freely and easily modeled using engrafting gestures. An engrafting gesture is defined as dragging stem part of one inflorescence diagram to any branch part of another inflorescence diagram. The engrafted inflorescence will adjust itself in position, size and direction to suit in the mapped branch. Fig.2 shows several compound inflorescence generation using engrafting gestures. As shown in Fig.2, our inflorescence diagram is 3D based consisting of spheres and line segments. A sphere stands for an individual flower and a line segment stands for a stem or a branch.

Inflorescence pattern usually contains 11 geometrical parameters as shown in Fig.3a. They are the stem root position P_0, stem height H_s, base length H_b, inter-node length H_d, branch length L_b, down angle α_v, rotation angle α_r (not shown in Fig.3), flower size R_b,

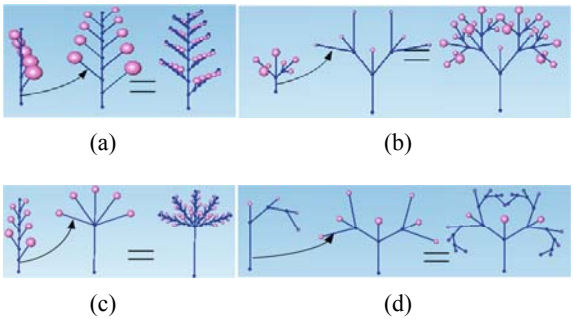


Fig.2 Compound inflorescence generation using engrafting gestures: (a) compound raceme; (b) compound dichasium; (c) raceme+umbel; (d) drepanium+dichasium

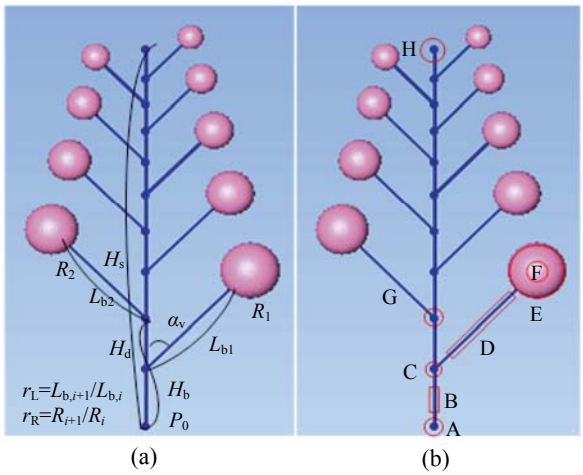


Fig.3 Definitions of the parameters and their editing gestures in inflorescence structural diagram. (a) Definitions of the parameters; (b) Editing gestures

neighbor branch length ratio r_L, neighbor flower size ratio r_R and the inflorescences degree N (which is taken as 9 in Fig.3a). Notice that usually r_L is set to be equal to r_R.

Traditional WIMP style manipulations in editing these geometrical parameters are very tedious to the users. Here we introduce nine dragging gestures to intuitively adjust those parameters directly onto inflorescences diagrams (as shown in Fig.3b). It should be pointed out that these editing gestures can be adapted to the whole inflorescence and also be adapted to sub-levels of inflorescence or individual components. Experiments show that even novice users can freely and quickly edit inflorescences geometric parameters to get a reasonable inflorescence structure.

Gesture A: When a user drags on region A, the system treats this gesture as editing the stem bottom

position in vertical direction. P_0 and H_b will be adjusted accordingly.

Gesture C: When a user drags on region C, the system treats this gesture as editing the base length. H_b will be adjusted accordingly, and the constraints are that the distance between the last branch's root and stem's root must be less than H_s, and H_b>0.

Gesture G: When a user drags on region G of any branch except the first, this gesture is considered to edit the inter-node length. H_d will be adjusted accordingly, and the constraints are that the distance between the last branch's root and stem's root must be less than H_s, and H_d>0.

Gesture D: When a user drags on region D of any branch, this gesture is considered to edit the branch length. L_b will be adjusted accordingly.

Gesture F: When a user drags on region F of any sphere, this gesture is considered to edit the branch length and down angle. α_v and L_b will be adjusted accordingly.

Gesture E: When a user drags on region E of any sphere, this gesture is considered to edit the flower size. R_b will be adjusted accordingly.

Gesture H: When a user drags on region H, this gesture is considered to edit the stem height. H_s will be adjusted accordingly, and the constraint is that H_s must be greater than the distance between the last branch's root and stem's root.

Gesture B: When a user drags stem regions except A, C, G and H (it is labelled as "B" in Fig.3b), this gesture is considered to translate the whole inflorescence diagram. P_0 will be adjusted accordingly.

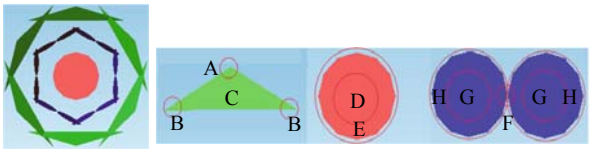
Gesture I (not shown in Fig.3b): When a user clicks the "branch rotating" button, the system will automatically switch to the top view, and any dragging on sphere will be treated as editing the rotation angle between the branch and the stem. α_r will be adjusted accordingly.

The geometrical modeling of inflorescence only contains stem or branch modeling. Stem or branch is implemented here as a swept surface. The section curve of the swept surface is a circle with the initial radius estimated by the diagonal length of bounding box of entities in current view. To construct the stem/branch, users only need to sketch a trajectory indicating the central axis of the swept surface. Users can adjust top and bottom radii using dragging gestures and can also use the replacement gesture to edit the central axis. Fig.4 shows an example of stem/branch modeling based on freehand sketching.

Fig.4 Creating and editing stem/branch based on freehand sketching

Individual flower modeling

In our system, different types of components in individual floral structural diagram are represented using different shapes and colors. Petal and calyx are expressed as green triangles, while the pistil is expressed as red circles, and the blue double circles represent the stamen. Fig.5a shows a floral diagram containing six petals/calyx, one pistil and four stamens.



(a) (b)

Fig.5 Floral diagram representation (a) and editing (b). Green triangles: petal and calyx; Red circles: pistil; Blue double circles: stamen

In the floral diagram, petal/calyx contains five parameters, namely the distance away from flower center, size, rotation angle, down angle and the number of petal/calyx. Stamen has the same five parameters as petal/calyx. Pistil has only two parameters: center position and size. Therefore, a floral diagram has 12 geometric parameters in total. Similar to inflorescences diagram editing, as shown in Fig.5b, we define nine dragging gestures to edit the floral diagram as follows:

Gesture A: When dragging on region A of the petal, the system treats this gesture as editing the distance between the petal and the flower center.

Gesture B: When dragging on region B of the petal, this gesture is considered to adjust the petal size.

Gesture C: When dragging on the petal region except regions A and B (it is labelled as "C"), the system treats this gesture as editing the rotation angle.

Gesture D: When dragging on region D of the pistil, it is considered to translate the whole flower diagram.

Gesture E: When dragging on the boundary region E of the pistil, it is considered to adjust the pistil size.

Gesture F: When dragging on center region F of the stamen, it is considered to adjust the distance between the stamen and flower center.

Gesture G: When dragging on region G of the stamen, it is considered to adjust the stamen's rotation angle.

Gesture H: When dragging on region H of the stamen, it is considered to adjust the stamen's size.

Gesture I: When a user clicks "down angle" button, the system will automatically switch to side view. A user can drag any region on petal/calyx or stamen; these gestures will be treated as editing the down angle of components.

The geometrical modeling of individual flower contains stamen modeling, pistil/receptacle modeling and petal/calyx modeling. Stamen is defined as an inflation object referred to in (Igarashi *et al*., 1999) representing anther, and a swept surface representing spermaduct. Pistil and receptacle are defined as a revolved surface created by two strokes. Petal and calyx are defined as a B-spline surface created by two boundary strokes. The central vein of petal is automatically generated by calculating an iso-parameter curve in the B-spline surface with the μ-parameter being 0.5. To edit those geometric components, we also defined several convenient gestures. Fig.6 shows the creation and editing based on freehand sketching.

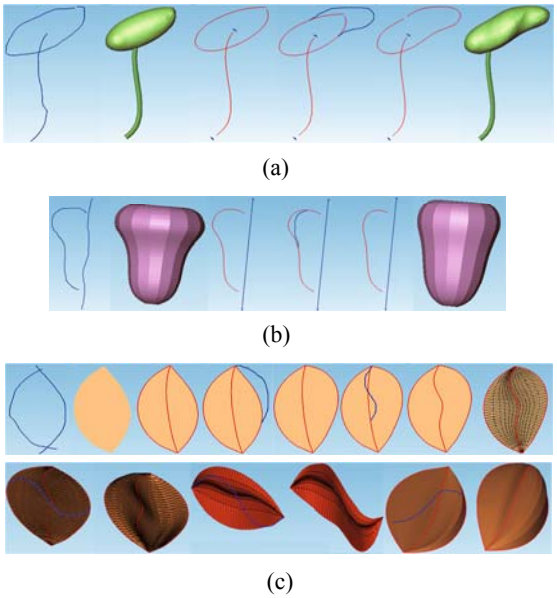


Fig.6 Creating and editing stamen, pistil and petal based on freehand sketching. (a) Stamen modeling; (b) Pistil/receptacle modeling; (c) Petal/calyx modeling

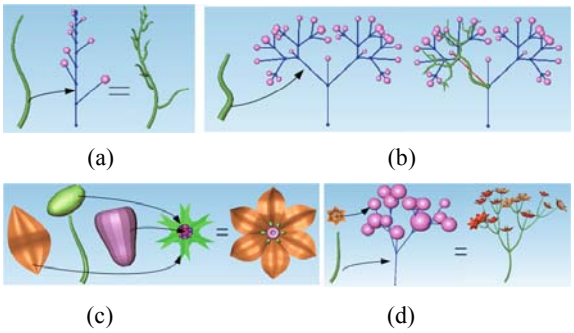


Fig.7 Mapping geometric components to structural diagrams. (a) Global stem mapping in inflorescence; (b) Local branch and its offspring mapping in inflorescences; (c) Mapping in individual floral diagram; (d) Mapping both individual flower and stem to inflorescence diagram

Mapping

Geometrical modeling of inflorescences and individual flower modeling are independent to the structural modeling. Now we combine them to construct realistic flower models by "mapping" them together. To facilitate the "mapping" operation, we use dragging gestures. The source geometrical object of mapping will be automatically translated, rotated and scaled to the destination structural diagram. The source object of dragging gesture must be matched with a specific destination object. For example, a user cannot drag a stem to any sphere in the inflorescence diagram which stands for an individual flower. Fig.7 shows samples of geometrical components mapping to structural diagrams.

It should be pointed out that since all geometric components in our system are expressed in triangular mesh, for an inflorescence which has lots of individual flowers and branches, the final mapped model could have a large number of triangles. It is necessary to simplify the model. Based on the method of levels of detail, here we simplify the mapped geometric components according to the size of destination object in the structural diagram during the mapping operation.

FORMING A SOLID SHAPE

Well-formed solid shape is an essential requirement for rapid prototyping application. This section focuses on detecting and correcting the wrong geometric relationships in flower models, and merging all components into one single waterproof solid shape.

Inflating petals and sepals

Since petal and sepal are defined as B-spline surfaces bounded by two curves, the tessellated mesh will be thin plate shape. Here we inflate the petal surface to the top and bottom independently, tessellate these two surfaces uniformly, and combine them together according to their topological relationships. The final combined mesh will be a solid shape.

Fig.8 shows our inflation algorithm. The raw petal surface will be inflated into two surfaces located at the top and bottom independently. The inflated shape is controlled by the parameter *r*: the larger the *r*, the thicker the petal shape. By default, *r* is set to 1.0. Notice that the coordinates of the boundary points in the petal surface will be maintained by our inflation algorithm, which means that the uniformly tessellated meshes of *TS* and *BS* in Fig.8 will be spatially connected together in the boundary region. Hence they can be easily sewed together using only topological operation without involving hole-filling. To get a smooth solid shape, before tessellation we should smooth the control network of *TS* and *BS* using Laplacian operators without changing the boundary points. Fig.9 shows an example using our petal inflation algorithm.

```
Procedure PetalInflation(S, r, TS, BS)
Input: S: the petal B-spline surface; r: the inflation ratio.
Output: TS: the inflated top surface; BS: the inflated bottom surface.
Local variables: m, n: the number of control points in U and V
direction of Surf, respectively; maxLen: max length of iso-curves in
V direction of Surf; maxDiff: maximum inflation distance of Surf
Begin
  For i=1 to m Do
    Length[i]=S.GetIsoCurve(i/(m−1.0),VDIR).Length();
  End For
  maxLen=max{Length[1], ..., Length[m]};
  maxDiff=S.BoundingBox().DiagonalLength()/40*r;
  For i=1 to m Do
    For j=1 to n Do
      u=(i−1.0)/(m−1.0);  v=(j−1.0)/(n−1.0);
      normal=S.EvaluateNormal(u, v);
      if (v<0.5) then
        dis=Length[i]/maxLen*maxDiff*v/0.5;
      else dis=Length[i]/maxLen*maxDiff*(1.0−v)/0.5;
      TS.SetCtrlPnt(i, j, S.CtrlPnt(i, j)+dis*normal);
      BS.SetCtrlPnt(i, j, S.CtrlPnt(i, j)−dis*normal);
    End For
  End For
End
```

Fig.8 Algorithm for the petal inflation

Fig.9 A sample of petal inflation. (a) Raw petal surface; (b) Inflation result of *r*=0.5; (c) Inflation result of *r*=1.0; (d) Inflation result of *r*=3.0; (e) Inflation result of *r*=6.0; (f) Section views of different inflation ratios

Branch merged with branch

Although the root of the sub-branch locats at the body of the parent branch, they are not really combined with each other in geometry and topology. Figs.10a and 10b describe some typical shape relations between the sub-branch and the parent branch. To form a solid shape, we need to merge them together.

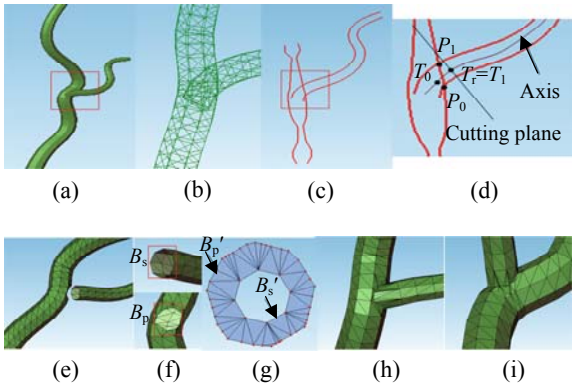


Fig.10 An example of branch-branch merging. (a) Raw shape; (b) Local details of incorrect geometry relationship; (c) Section curves; (d) Finding a cut plane; (e) Cutting and trimming; (f) Local details of cutting and trimming boundaries; (g) CDT result; (h) 3D shape recovered from CDT topology and merged together; (i) Another view of (h)

The merging algorithm is as follows:

Step 1: Calculate a section plane *P*. *P* is defined by two line segments in the inflorescence diagram representing the sub-branch and the parent branch.

The normal of P is the cross product of these two line directions. And P passes through the root position of the sub-branch line segment.

Step 2: Calculate the section curves of the sub-branch and the parent branch associated with P (Fig.10c). Calculate the intersection points between parent branch section curves and sub-branch section curves. Search the nearest points at the axis of sub-branch with those intersection points. Choose the point with the largest parameter at the axis as the reference point T_r. Evaluate the tangent direction of T_r at the axis, and finally get a plane $C(T_r, \boldsymbol{N}_r)$, $\boldsymbol{N}_r$ is the tangent vector. In Fig.10d, parent branch section curves intersect with sub-branch section curves, and we get two intersection points P_0 and P_1. The nearest points to P_0 and P_1 at the axis are T_0 and T_1, respectively. It is obvious that $T_1>T_0$, so $T_r=T_1$.

Step 3: Cut the root region of the sub-branch mesh using plane C and we get a section boundary polygon B_s. Then use B_s to project onto the parent branch and trim a hole at the facade of the parent branch. The projecting direction is N_r and the trimmed boundary polygon is B_p. Figs.10e and 10f show the results after cutting and trimming.

Step 4: Project B_p onto the plane C vertically, the projected polygon is named as B_p'. Scale B_s by its barycenter until it is completely contained by B_p'. The polygon after scaling is named as B_s'. Use constrained Delauny triangulation (CDT) (Shewchuk, 1996) to triangulate B_p' and B_s' (Fig.10g).

Step 5: According to the topological relationships (B_p and B_s matched with B_p' and B_s', respectively), we can easily build a spatial mesh from the CDT plane mesh. The vertices of spatial mesh are the points of B_p and B_s and the topology of spatial mesh is totally the same with CDT plane mesh. We use the spatial mesh to fill the gap between the sub-branch and the parent branch. Finally, the sub-branch will be merged into the parent branch successfully and the local shape is waterproof now. Figs.10h and 10i show two views of the merged results.

Merging spermaduct with anther

Anther is defined as an "extrusion" shape, and spermaduct as a swept surface. Stamen cannot be sewed directly because spermaduct and anther are not really connected with each other geometrically (Fig.11a). Using an algorithm similar to the branch-branch merging, we can easily and quickly merge anther with spermduct. Fig.11 shows an example of merging spermaduct and anther.

Merging interiors of an individual flower

Here we take a similar approach to automatically merge stamen, petal/calyx and pistil with receptacle. First, we cut a hole on the bottom regions of stamen, pistil and petal/calyx. The cutting is easy to implement with a small topological operation because the spermaduct is a swept surface, the pistil is a revolved surface, and the inflated petal has a strictly regular shape. Those boundaries will then be projected to the base plane of the receptacle top. Fig.12d shows the projecting result. It should be noticed that the boundaries of stamen, pistil and petal/calyx should be totally contained by the boundary of the receptacle. Hence we need to relocate those boundaries to satisfy this rule, and to avoid collision with each other during the relocation. Fig.12e shows the result of relocated boundaries. We then triangulate those boundaries with the CDT method. Finally we combine all components onto receptacle using the CDT mesh as the bridge.

Fig.13 shows an example resulted from our solid flower modeling system.

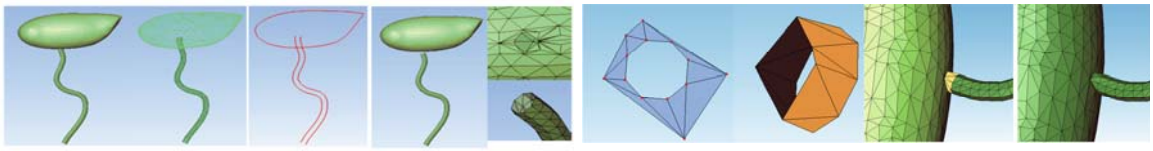


Fig.11 An example of spermaduct-anther merging. (a) Raw stamen; (b) Section curves; (c) Results of stamen after cutting and trimming; (d) CDT result; (e) Recovered 3D shape from the CDT mesh; (f) Results of bridging spermaduct and anther, and merging them together

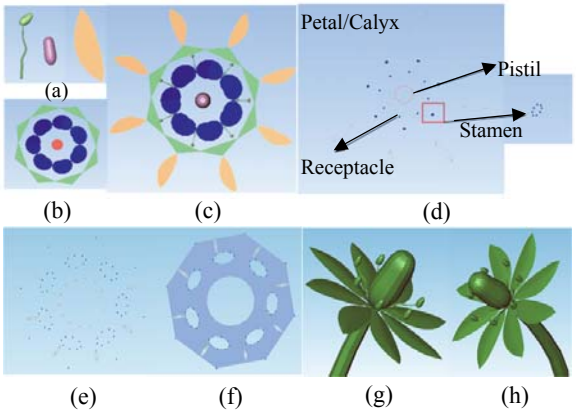


Fig.12 An example of merging interiors of individual flowers. (a) Raw components: stamen, pistil, petal/calyx; (b) Floral diagram; (c) Mapping result; (d) Boundaries of each component; (e) Results after relocating; (f) CDT mesh; (g) Merging result; (h) Another view of (g)

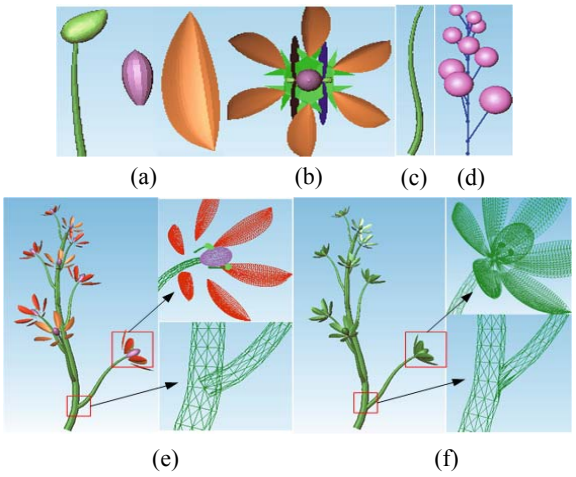


Fig.13 An example of solid flower modeling. (a) Geometric components: stamen, pistil, petal; (b) Mapped floral diagram; (c) Stem; (d) Inflorescence; (e) Initial modeling result before forming a solid; (f) Final solid shape

CONCLUDING REMARKS

In this paper, we present a sketch-based method for modeling vivid flowers. Based on (Ijiri *et al.*, 2005)'s method, we separated flower modeling procedure into two parts: individual floral diagram modeling and inflorescence modeling. We introduced a rich set of implicit editing gestures to allow the user freely editing structure parameters. In our system, all 3D geometrical components are created and edited by freehand sketching. The final step of modeling is to automatically merge all 3D components into a waterproof mesh. Our presented merging algorithm is simple and efficient. Initial experiments show that our approach is promising. Novice users can create vivid flower model easily and quickly, and the final model can be printed onto real flower toy or decoration directly.

References

Bell, A.D., 1991. Plant Form: An Illustrated Guide to Flowering Plant Morphology. Oxford University Press.

Boudon, F., Prusinkiewicz, P., Federl, P., Godin, C., Karwowski, R., 2003. Interactive design of bonsai tree models. *Computer Graphics Forum*, **22**(3):591-599. [doi:10.1111/1467-8659.t01-2-00707]

Cohen, J., Markosian, L., Zeleznik, R., Hughes, J., Barzel, R., 1999. An Interface for Sketching 3D Curves. Proc. Symp. on Interactive 3D Graphics, p.17-21. [doi:10.1145/300523.300655]

Eggli, L., Hsu, C., Elber, G., Bruderlin, B., 1997. Inferring 3D models from freehand sketches and constraints. *Computer-Aided Design*, **29**(2):101-112. [doi:10.1016/S0010-4485(96)00039-5]

Igarashi, T., Matsuoka, S., Tanaka, H., 1999. Teddy: A Sketching Interface for 3D Freeform Design. Proc. ACM SIGGRAPH, p.409-416.

Ijiri, T., Owada, O., Okabe, M., Igarashi, T., 2005. Floral Diagrams and Inflorescences: Interactive Flower Modeling Using Botanical Structural Constraints. Proc. ACM SIGGRAPH, p.720-726. [doi:10.1145/1186822.1073253]

Lipson, H., Shpitalni, M., 1996. Optimization based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design*, **28**(8):651-663. [doi:10.1016/0010-4485(95)00081-X]

Michalik, P., Kim, D.H., Bruderlin, B.D., 2002. Sketch- and Constraint-based Design of B-spline Surfaces. Proc. Solid Modeling, p.297-304.

Noser, H., Rudolph, S., Stucki, P., 2001. Physics-Enhanced L-systems. Proc. Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision, **2**:214-221.

Okabe, M., Owada, S., Igarashi, T., 2005. Interactive Design of Botanical Trees Using Freehand Sketches and Example-based Editing. Eurographics, p.487-496.

Pentland, A., Kuo, J., 1989. The Artist at the Interface. Vision Science Technical Report.

Prusinkiewicz, P., Lindenmayer, A., 1990. The Algorithmic Beauty of Plants. Springer-Verlag, New York.

Prusinkiewicz, P., James, M., Mech, R., 1994. Synthetic Topiary. Proc. 21st Annual Conf. on Computer Graphics and Interactive Techniques, p.351-358. [doi:10.1145/192161.192254]

Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J.D., Kang, S.B., 2006. Image-based Plant Modeling. Proc. ACM SIGGRAPH, p.599-604.

Shewchuk, J.R., 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delauny Triangulator. Proc. First Workshop on Applied Computational Geometry, p.124-133.

Tanaka, T., Naito, S., Takahashi, T., 1989. Generalized symmetry and its application to 3D shape generation. *The Visual Computer*, **5**(1-2):83-94. [doi:10.1007/BF01901484]

Zeleznik, R.C., Herndon, K.P., Hughes, J.F., 1996. SKETCH: An Interface for Sketching 3D Scenes. Proc. ACM SIGGRAPH, p.163-170.