EasyToy: Plush Toy Design Using Editable Sketching Curves

Yong-Jin Liu ▪ *Tsinghua University*

Cui-Xia Ma ▪ *Chinese Academy of Sciences*

Dong-Liang Zhang ▪ *Liveforce*

In EasyToy, editable sketching curves combine the advantages of free-form strokes and the controllability of B-spline curves. Users can continuously edit each curve to refine designs. A set of simple tools lets users easily construct sophisticated toy models comparable to those that professional systems produce.

In the past decade, research on 3D graphics modeling techniques has flourished. With the pervasiveness of PCs and smart user interfaces, 3D model design now can take place on both professional systems running on workstations and user-centered systems running on conventional PCs.

One industrial application of 3D models is to manufacture customized plush toys. State-of-the-art commercial systems include Rhino, Maya, 3ds Max, and SolidWorks. Although these systems let users precisely construct sophisticated models, they require professional skills. Teddy was the first system to provide a gesture-based interface that lets nonprofessional users, especially children, design 3D free-form models easily.[1] Teddy recently was extended to the Plushie system, which lets users construct a 3D toy model and a 2D pattern of developable pieces in an interweaved fashion.[2] Teddy and Plushie are easy to learn and natural to use, but the generated models are much simpler than those that professional systems produce.

Sketching is a fundamental technique that can naturally express a designer's intent. In systems for designing 3D models, sketching can involve specific different types of strokes,[3] free-form strokes,[1] or 3D control curves.[4] In addition, researchers have proposed gesture-based interfaces that employ different geometric processing techniques. However, gesture-based systems can model relatively few gestures because of the human memory's limited capacity. When a system uses too many gestures, users have difficulty remembering the gesture operations exactly, so their cognitive load increases. Error-prone behavior also restricts applications of context-based sketching systems. So, current gesture-based systems typically produce much less complex models than professional WIMP (window, icon, menu, pointing device) systems.

To overcome this problem, we developed EasyToy, an interactive toy design system that employs *editable sketching curves*. This technique combines the advantages of the natural expression of free-form strokes and the controllability of B-spline curves. Users can refine their concepts by continuously editing these curves. Although EasyToy's learnability and usability are similar to those of Teddy, it can construct models whose complexity is comparable to models from professional systems (see Figure 1).

To enable simple, intuitive 3D plush toy model design, EasyToy adopts mesh models and a small set of simple design tools. EasyToy's workflow consists of a set of geometric objects and operation commands like those used in 2D and 3D modeling and editing systems. Users can sketch curves representing parts of the toy displayed; these curves have some degree of fuzziness, which users can clarify later. Users don't need to switch excessively among menus, buttons, and the keyboard; they can focus on the design task instead of the tools. These distinct characteristics also make EasyToy well suited for industry.

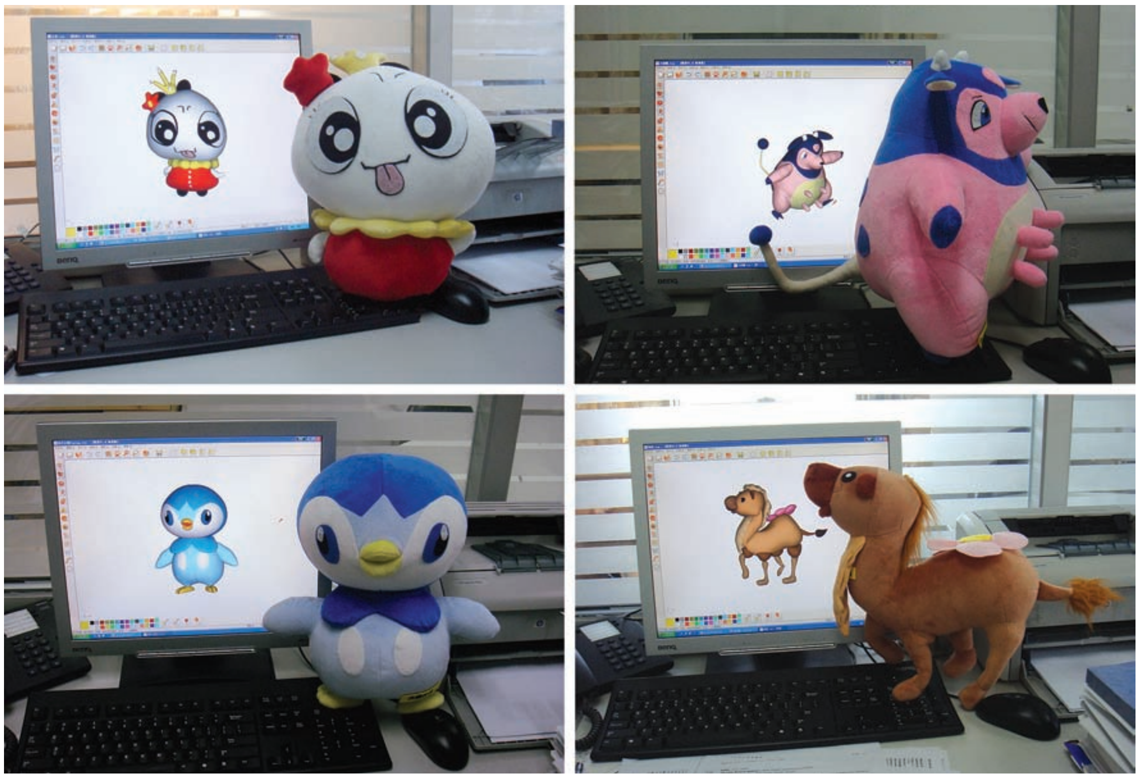

Figure 1. Toy models in different styles that novices created using EasyToy. Using editable sketches, users can easily construct sophisticated toy models in EasyToy.

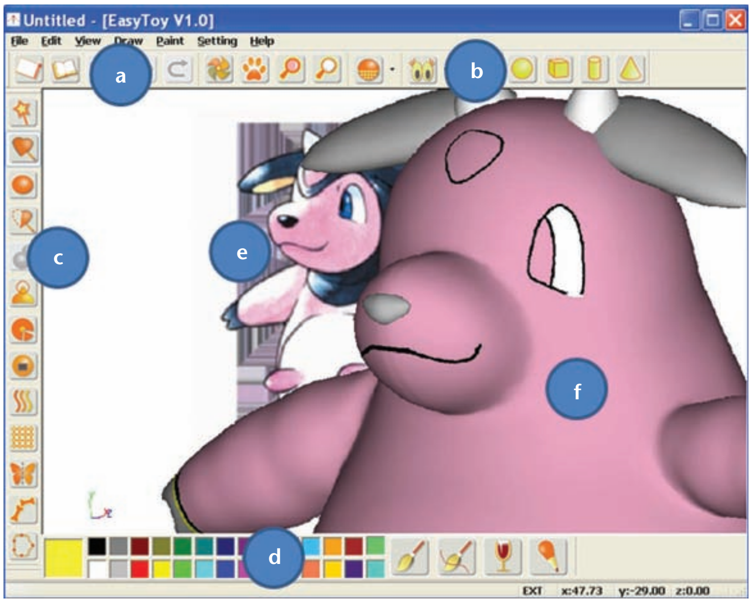
Untitled - [EasyToy V1.0]
File Edit View Draw Paint Setting Help
a
b
c
d
e
f
EXT x:47.73 y:-29.00 z:0.00

Figure 2. The EasyToy environment. (a) The navigation toolbar. (b) The geometric-primitive toolbar. (c) The geometric-modeling toolbar. (d) The painting toolbar. (e) The background, showing a 2D reference drawing (hand-created illustrations or photographs). (f) A 3D toy model (the final model appears in the upper right of Figure 1). All the toolbars are dockable.

EasyToy's Framework

Figure 2 shows EasyToy's interface, illustrating the trade-off between expressiveness and naturalness. The interface has four toolbars, all of which are dockable. The toolbars' default locations are at the interface's top, left, and bottom. Users can perform all system operations using the mouse or keyboard shortcuts. They can select icons to execute all operations, without using compound menus or multilayer panels.

The *navigation toolbar* (see Figure 2a) includes manipulation, viewing, and rendering operations. Manipulation operations activate different functions. Users employ the mouse to translate on, rotate about, zoom in on, and zoom out of the object model. The viewing operations set up viewing directions, including front view, top view, and back view. The rendering operations set different rendering modes, including wireframe, shaded, and translucent. (The translucent mode is important when users want to create 3D objects from a background 2D reference drawing.)

The *geometric-primitive toolbar* (see Figure 2b) includes frequently used shapes such as a sphere, cylinder, cube, and cone.

The *geometric-modeling toolbar* (see Figure 2c) contains all the necessary modeling operations: Boolean, symmetrization, smoothness, deformation, and painting. Every operation is mapped to an icon so that users can easily understand the icons' meanings. We selected these operations from dozens of candidate modeling techniques, guided by an informal user study by Takeo Igarashi and his colleagues.[1] EasyToy offers a trade-off between system usability and the design power of more sophisticated toy-modeling systems.

The *painting toolbar* (see Figure 2d) includes a color palette and operations such as selecting a color, filling in a selected area with a color, drawing an editable curve with a color, and drawing a stroke with a color.

The central window is used both for 2D sketching and 3D model creation (see Figures 2e and 2f). The system uses OpenGL to set up a 3D environment and maintain a projection plane associated with the current eye or camera position. It also uses parallel projection. EasyToy displays 2D sketching on the projection plane—that is, the screen plane.

How EasyToy Works

Here we describe how EasyToy works, with some simple functions.

Creating 3D Shapes from Planar Sketching Curves

Users draw 2D planar sketching curves on a screen plane by tracing the mouse's movement. Users can modify the initial draft and refine it by continuous interaction. For this purpose, we approximate the polyline by a B-spline curve of degree three:

$$C(u) = \sum_{i=0}^{n} N_{i,3}(u) P_i, 0 \le u \le 1,$$

where $N_{i,3}$ is the basic function of degree 3 and the knot vector $U = \{0, 0, 0, 0, u_4, u_5, \ldots, u_n, 1, 1, 1, 1\}$. The curve can pass through control points (see Figure 3a). To achieve this, we set the knots to $0 < u_4 < u_5 < \ldots < u_n < 1$ and the new control points, Q_i on curve C, to $C(u_i)$, where $i = 3, 4, \ldots, n + 1$. The relation $Q = AP$ of control vectors Q and P holds with an $(n - 1) \times (n + 1)$ sparse matrix A. To invert A, we add two constraints of tangent vector preservation at the endpoints. By inserting and removing knots, users can add or delete control points on the curve (see Figure 3b) by clicking the left and right mouse buttons, respectively.

Given an edited closed planar curve C, the system extrudes a round object by using C as the silhouette. Unlike Teddy's extrusion method, EasyToy uses an ellipse with the radius $(1, r)$ to interactively specify the object's thickness r. If the user sketches a circular curve and r is larger or smaller than 1 (see Figure 3c, left), the algorithm generates an ellipsoid (see Figure 3d, left). If r is 1 (see Figure 3c, right), the algorithm generates a roughly spherical object (see Figure 3d, right).

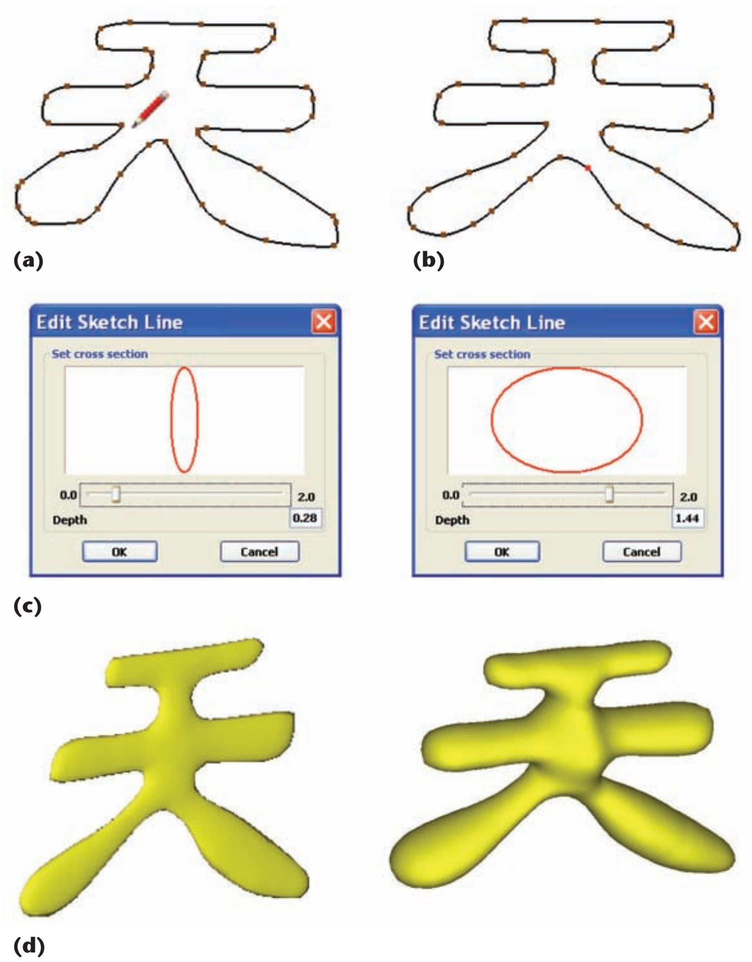

Figure 3. Designing a solid Chinese character. (a) The initial sketched curve. (b) Curve modification. (c) Two thickness settings. (d) The characters generated with those thicknesses.

Smoothness Operations

The smoothness tools include mesh simplification, mesh smoothing, and mesh refinement. These tools can be used either globally or locally. With the local method, the user first sketches a closed curve on an object surface to enclose a selected region. For mesh simplification, we use quadratic error metrics.[5] For mesh smoothing, we use the umbrella operator[6] because it can smooth the shape and produce a uniform mesh edge length, which is desired in round plush toy models. For mesh refinement, we use 1:4 split rules. In local refinement, the triangles adjacent to refined regions are split by 1:2 rules.

Boolean Operations

EasyToy includes three Boolean operations. The copy operation duplicates a selected object. The cut operation is similar to that of the Teddy system. The merge operation combines two intersected objects into a two-manifold object. B-rep models aren't always closed under Boolean set operations, so the merging operation is ill posed.

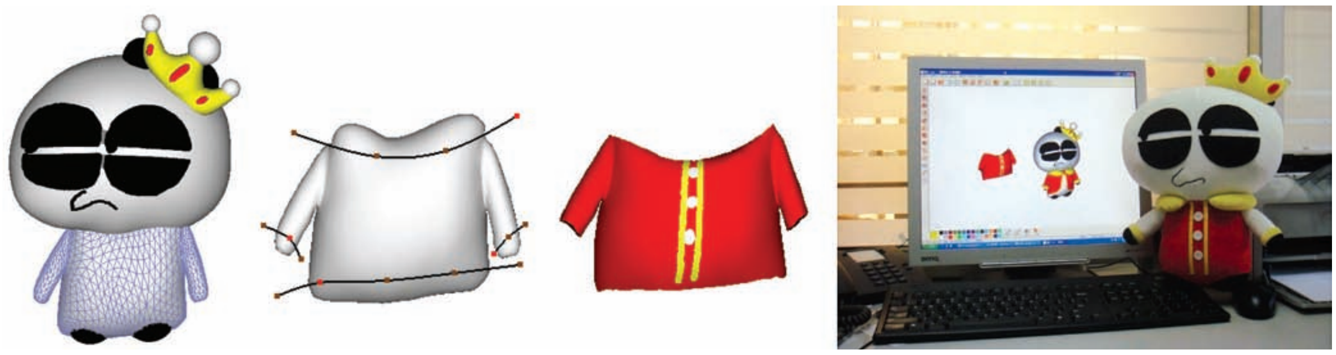

Figure 4. Copying and cutting to make tight clothes. (a) Selecting and copying a body. (b) Cutting the body with four strokes. (c) Painting the cloth. (d) The manufactured model.

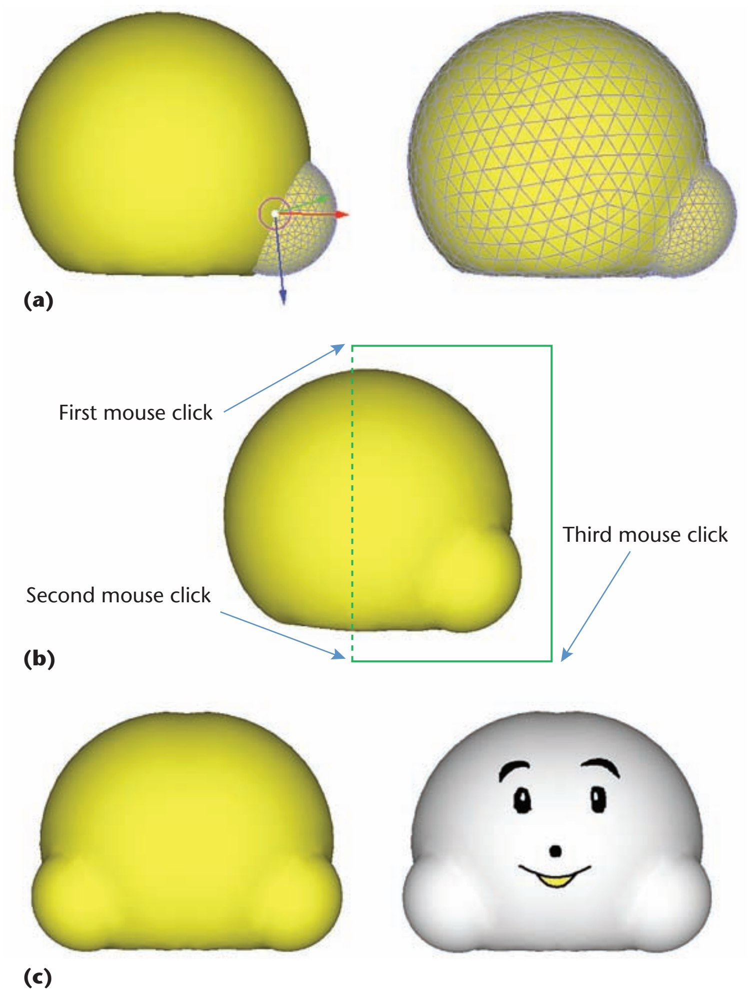

Figure 5. A toy head design using merging, 3D symmetrization, and painting. (a) Locating and merging two spheres. (b) The mouse clicks involved in selecting the symmetric plane. (c) 3D symmetrization and painting of the selected part.

In EasyToy, the merge operation functions interactively and conservatively. After a user places two intersected objects in space, he or she draws a stroke on them near the intersection area. The stroke's endpoints indicate that the two objects are to be merged.

Let one endpoint be projected onto a triangle *T* in an object *A*. Starting from *T*, EasyToy searches all triangles in *A* breadth-first to find an initial triangle *ITri* that's intersected by the other object *B*. Initialized by the intersection in *ITri*, the intersection between *A* and *B* is traced. EasyToy performs the tracing conservatively, with a tolerance of 10^{-3} in floating-point arithmetic. If it detects nonmanifoldness, such as the contact of two triangles from *A* and *B*, it terminates tracing and displays the error message "Wrong placement" in a pop-up message box near the mouse. In this case, EasyToy displays a message near the mouse cursor suggesting that the user move one of the objects. Figure 4 and Figure 5a show two examples using Boolean operations.

Symmetrization

EasyToy uses simple, effective 3D symmetrization. The user first rotates the object to a suitable position so that the symmetric plane will be perpendicular to the projection plane. Next, the user selects two points in the projection plane to define the symmetric plane and a third point to create the selection window (see Figure 5b). Boolean operations help achieve 3D symmetrization (see Figure 5c).

Deformation

Deformation operations in EasyToy include local disk region deformation, articulate deformation, and bounding-box deformation. All deformation operations are intuitive.

Local disk region deformation. In this operation, the user first selects a vertex on the object serving as a disk center *O*. For easy sketch-based deformation, EasyToy displays the normal at *O* with an arrow (see Figure 6a, top left). All vertices falling in the geodesic distance *r* with respect to *O* appear in a different color. Users can modify *r* by pressing the up and down arrow keys on the keyboard (see Figure 6a, bottom right). After the user specifies *O* and *r*, he or she drags *O* to extrude or intrude the local region (see Figure 6a, top right, and Figure 6a, bottom left). The normal of O determines the deformation's direction. In the screen plane, the line perpendicular to the normal partitions the plane into two halves. If the dragging direction is in the positive half-plane, EasyToy makes an extrusion. If the dragging direction is in the negative half-plane, the program makes an intrusion. Dragging also determines the deformation's magnitude. For a smooth deformation, the vertex at distance $r' < r$ to O has a magnitude with a degraded coefficient:

$$c\left(r'\right)=\left(\left(\frac{r'}{r}\right)^2-1\right)^2, \ r' \leq r \, .$$

Articulate deformation. This operation projects all vertices v of a 3D object onto the projection plane, pv. Upon a rigid motion, we define the projection plane as the xy plane, and the coordinates of v and pv are (x, y, z) and (x, y), respectively. The articulate deformation drives the selected vertices in the (x, y) coordinates, and the z coordinate is unchanged in 3D space. On the projection plane, the user first selects a point on the object as a joint (see Figure 6b, left). The user then selects the second point, which forms a bone with the first point (see Figure 6b, center). Then the user can move the second point. EasyToy moves all the selected 2D vertices pv according to the rotation defined by the second point's motion around the first point (see Figure 6b, right). This deformation implements a vertex-blending method.[7]

Bounding-box deformation. Similar to articulate deformation, this operation also works with the 2D projection pv of 3D vertices on the projection plane. Unlike articulate deformation, it modifies the object by specifying a bounding box that can be equipped with different orientations. Bounding-box deformation uses a free-form-deformation method[8] with a simplified 2D technique. The user first specifies an axis by selecting two points (see Figure 6c, left). Then EasyToy automatically generates the object's bounding box along the orientation indicated by the axis (see Figure 6c, center). By dragging the control points distributed at the bounding box's boundary, the user deforms the enclosed region in the box accordingly. Users can drag, add, or delete these control points (see Figure 6c, right).

3D Painting

EasyToy employs parameterization-insensitive 3D painting, which has these advantages:

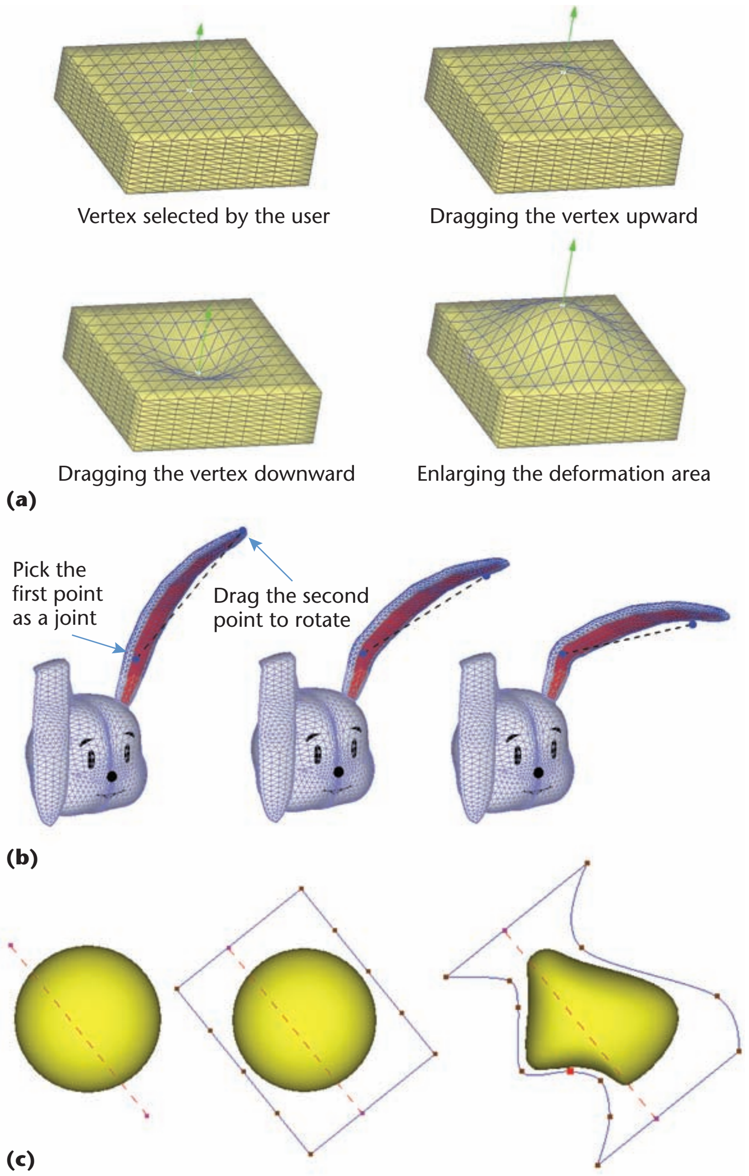
Vertex selected by the user
Dragging the vertex upward
Dragging the vertex downward
Enlarging the deformation area
(a)
Pick the first point as a joint
Drag the second point to rotate
(b)
(c)

Figure 6. Three types of deformation in EasyToy: (a) local disk deformation, (b) articulate deformation, and (c) bounding-box deformation. The three deformation tools provide a simple, intuitive technique with which users can create sophisticated models.

- Round toy models usually are nondevelopable. Parameterizing round toy models into a few large charts in a 2D texture plane might introduce discontinuities and stretches. Parameterization-insensitive painting avoids these artifacts.
- Users can continuously edit an object's geometry and topology.

Parameterization-insensitive 3D painting supports both editable sketching curves (see Figure 7a, second icon) and strokes (see Figure 7a, fourth icon) on objects. Users can control the sketching curves' width by pressing the up and down arrow keys. They can modify the curves' shape by moving, adding, or deleting control points.

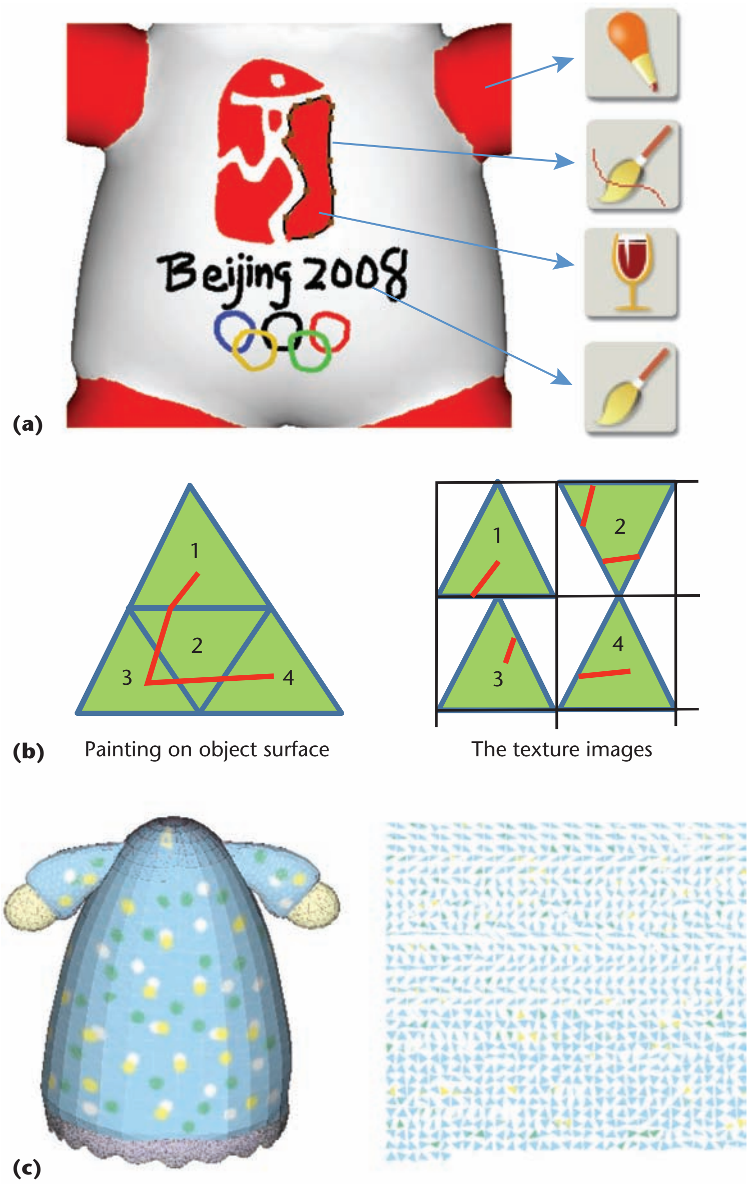

Figure 7. 3D Painting in EasyToy. (a) With the painting tools, users can pick a color (first icon), paint with an editable sketching curve (second icon), fill a local region with a color (third icon), or paint with strokes (fourth icon). (b) EasyToy allows parameterization-insensitive painting. (c) The program created this petticoat model; the texture image is on the right.

For each triangle passed by the sketching curve, EasyToy creates an independent texture image fragment (see Figure 7b). The parameterization-insensitive 2D texture image consists of fragments of all the painted triangles. EasyToy applies a heuristic algorithm[9] to optimize the packing of convex polygons in order to minimize a textured image's size. A textured image's size depends only on the number of painted triangles.

Most traditional texture mapping parameterizes the entire object, disregarding the painted area. Because EasyToy uses a parameterization-insensitive technique, storing and rendering the textured image is more efficient. Figure 7c shows a model of 14,550 triangles and fine painting details displayed uniformly on the object surface. This model's file size in EasyToy is 2,080 Kbytes; a similar model in 3ds Max is 2,867 Kbytes.

EasyToy's painting tools include two additional functions. Users can acquire a color from a background image or from an object surface (see Figure 7a, first icon). Given a closed local area on an object's surface, users can fill the area with a selected color (see Figure 7a, third icon). For triangles filled with a single color, EasyToy stores one single-triangle texture fragment.

2D Pattern Generation

To generate 2D patterns from 3D toy models, EasyToy uses editable sketching curves to interactively segment the 3D object (see Figure 8d). For each segmented patch, EasyToy constructs a 2D pattern using a physically based unwrapping algorithm similar to the research of Charlie Wang and his colleagues.[10] The algorithm operates in three phases. First, given a 3D mesh piece M_{3D}, the algorithm constructs an initial planar mesh M'_{2D} with the same connectivity of M_{3D} by flattening triangles in M_{3D}, one by one, into a plane. Second, it deforms M'_{2D} in the plane with the constraint of an overlap penalty to release the energy. Finally, when the algorithm attains the rest state M_{2D}, the energy is minimized. Users can control flattening accuracy by cutting the mesh. The cutting line is suggested by the interface, which seeks the greatest gradient-descending direction in the energy field of 2D patterns.

Figure 8 shows the complete process of toy model design, 2D pattern generation, and manufacturing.

User Experiences

To aid users in creative work, design tools must be compatible with workplace practices and enable users to use their traditional methods. We aim to develop an interactive 3D toy modeling system that enables a broader audience of novices who aren't familiar with 3D modeling to create customized plush models with easy-to-use, natural-feeling tools.

To test EasyToy's learnability and usability, we performed empirical research in a university general-elective course. We formally observed 76 users over a two-year period. The users comprised 24 females and 52 males between 17 and 24 years old. Of the users, 95 percent had no knowledge of 3D graphics modeling. The remaining 5 percent had used 3ds Max or other professional systems to perform modeling. All users were from schools of the humanities and social sciences, materials science, architecture, chemistry, economics, or management. They didn't necessarily have mathematical or programming backgrounds.

Learnability and Usability

Each user received 30 minutes of training on typical EasyToy operations. We then had each user design a toy any way he or she liked, with no limitations on the toy's appearance. The users generally created their own models at home within one week. Each user spent about one to three hours, not including preparation of the 2D illustrations. The users created more than 100 3D models. Each user received one plush toy that he or she had designed. We recorded approximately 40 pages of remarks from the users—for example:

> The interface is clear and simple, and is really good. It is not difficult to master the system.
>
> There are powerful operations for 3D modeling, especially, users can create the appropriate 3D modeling in terms of several simple tools. For example, the deformation operations can modify the shape conveniently, which I often use.
>
> I think it is an incredible system, with very simple and convenient tools. I [have previously] used the powerful Photoshop and 3ds Max systems. And it is also difficult to create sophisticated graphics, though I have learned them with much time, but EasyToy helped me figure out the problem. I mastered EasyToy so fast only after teacher's simple introduction and reading the guide.

To evaluate EasyToy's usability, we performed another study with six users at the same university (see Figure 9): four novices and two skilled users (who could use 3ds Max fluently). We gave them all the same task: given a 2D illustration (see the first image in Figure 9), create a 3D toy model using EasyToy, 3ds Max, and SmoothTeddy.[11] Before the experiment, we provided training on each system.

Afterward, the users gave their subjective opinions of the three systems, in terms of time spent and the models' design quality.

Using 3ds Max, two novices spent eight to 10 hours, one spent four to seven hours, one spent more than 10 hours, and the two skilled users spent one to three hours. Using EasyToy, two users spent less than one hour, three spent one to three

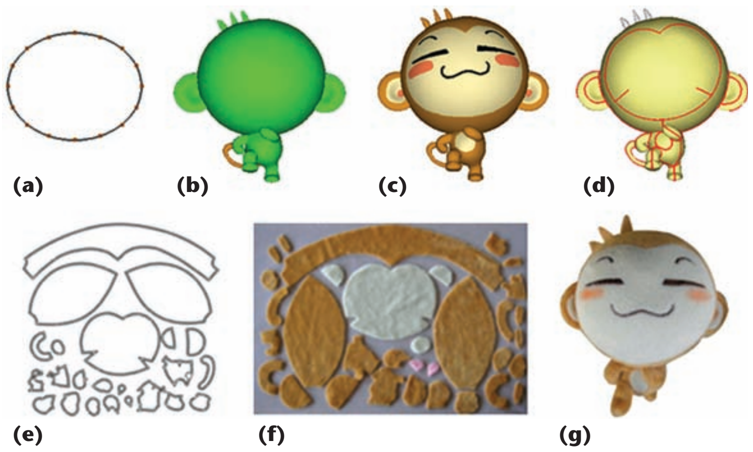
Figure 8. Toy model design and manufacturing: (a) Modeling by editable sketches. (b) The geometric model. (c) Painting. (d) Interactive specification of sewing lines by editable sketching curves. (e) 2D pattern generation. (f) Trimmed cloth material. (g) The manufactured model.

hours, and one spent four to seven hours. Using SmoothTeddy, two users spent less than one hour, two spent one to three hours, and two spent four to seven hours. Figure 10 includes more examples created by more novices in a university general elective course. Generally, EasyToy took less time because 3ds Max uses complicated compound menus and SmoothTeddy can't create some sophisticated parts easily. Such parts include the facial seams (the first row in Figure 10), the crown and the cloth (the second row in Figure 10), the sleeve straps (the last row in Figure 10), and the detailed color painting (see Figure 7).

For 3ds Max, four users ranked the design quality as excellent and two ranked it as good. For EasyToy, one user ranked the design quality as excellent and five ranked it as good. For SmoothTeddy, one user ranked the design quality as good, four ranked it as average, and one ranked it as poor.

The users also evaluated EasyToy's painting and editable-sketching functions, which we regard as particularly important. Four users ranked both functions as necessary, and two ranked them as important. No one ranked them as normal, optional, or unnecessary.

EasyToy's Advantages

Most users indicated that EasyToy had improved such design tools' usability and learnability. Generally, they felt that "the simple and powerful modeling tools had increased their efficiency and satisfaction with using the system under an informal design situation." From the observations of the 76 users over a two-year period, we found that 40 followed their traditional habit of using hand-created illustrations or scanned photographs. The editable sketching curves in EasyToy helped users perform paper-and-pencil-like 2D

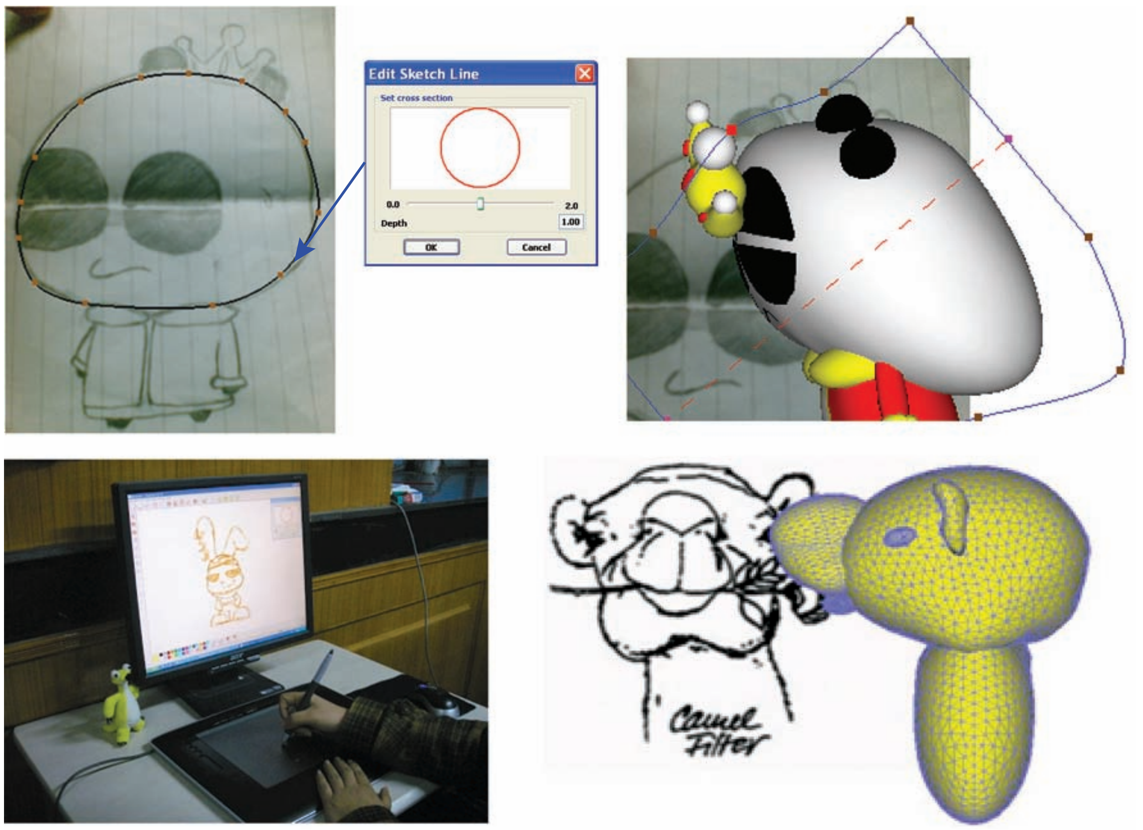

Figure 9. Paper-and-pencil-based 2D and 3D modeling by novices in EasyToy, using editable sketching curves. EasyToy can fluently address users' design intent without the limitations and interruptions that appear often during the modeling processes in state-of-the-art professional systems.

and 3D modeling (see Figures 9 and 10). Users can express their design intent without any distractions and interruptions.

EasyToy's Limitations

EasyToy uses the representation of both a 2D projection plane and a 3D modeling space. It functions effectively by supporting both planar and spatial sketching curves. However, some novices had difficulty understanding how to rotate and translate the model in space, especially when relating to the projection plane. We plan to perform a more detailed study of user experiences with sketch-based interactive design, under the guidance of user models based on cognitive science and the new interaction technologies of sketch-based interfaces.

Providing a better user experience during 3D modeling is difficult but important. Most systems don't pay much attention to continuous interaction, which means that most interactive behaviors are discrete.[12] Designing 3D modeling systems requires learning how to provide continuous interaction to prevent users from interruptions and how to improve the user experience. Our next study will focus on these aspects.

The emergence and development of sketch-based interfaces and context-awareness technologies will lead to new design methods and interactive technologies that will aid 3D modeling processes. In particular, developers will be able to incorporate sketch-based interfaces and context awareness into tools that enable users to transform dynamic design contexts into models naturally and effectively. In future research, we will consider adding multimodal interactions to a sketch-based interface to facilitate modeling and reusing existing models by 3D model retrieval with a partial matching method.

For more information about EasyToy, visit www.crefun.com/en.

Acknowledgments

We thank Charlie Wang at the Chinese University of Hong Kong for his discussion and comments on this research. This research was supported by China's 973 Program under project 2006CB303105 and by China's National Natural Science Foundation under projects 60603085, 60736019, and 60970099.

References

1. T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A

Sketching Interface for 3D Freeform Design," *Proc. Siggraph*, ACM Press, 1999, pp. 409–416.

2. Y. Mori and T. Igarashi, "Plushie: An Interactive Design System for Plush Toys," *ACM Trans. Graphics*, vol. 26, no. 3, article 45.
3. R. Zeleznik, K. Herndon, and J. Hughes, "Sketch, An Interface for Sketching 3D Scenes," *Proc Siggraph*, ACM Press, 1996, pp. 163–170.
4. A. Nealen et al., "Fibermesh: Designing Freeform Surfaces with 3D Curves," *ACM Trans. Graphics*, vol. 26, no. 3, 2007, article 41.
5. M. Garland and P. Heckbert, "Surface Simplification Using Quadric Error Metrics," *Proc. Siggraph*, ACM Press, 1997, pp. 209–216.
6. L. Kobbelt et al., "Interactive Multi-resolution Modeling on Arbitrary Meshes," *Proc. Siggraph*, ACM Press, 1998, pp. 105–114.
7. J. Lander, "Skin Them Bones: Game Programming for the Web Generation," *Game Developer Magazine*, May 1998, pp. 11–16.
8. T. Sederberg and S. Parry, "Free-Form Deformation of Solid Models," *Proc. Siggraph*, ACM Press, 1986, pp. 151–160.
9. R. Grinde and T. Cavalier, "A New Algorithm for the Minimal-Area Convex Enclosure Problem," *European J. Operational Research*, vol. 84, no. 3, 1995, pp. 522–538.
10. C. Wang, C. Smith, and M. Yuen, "Surface Flattening Based on Energy Model," *Computer-Aided Design*, vol. 34, no. 11, 2002, pp. 823–833.
11. T. Igarashi and J.F. Hughes, "Smooth Meshes for Sketch-Based Freeform Modeling," *Proc. ACM Siggraph 2003 Symp. Interactive 3D Graphics*, ACM Press, 2003, pp. 139–142.
12. T. Ju, Q.Y. Zhou, and S.M. Hu, "Editing the Topology of 3D Models by Sketching," *ACM Trans. Graphics*, vol. 26, no. 3, 2007, article 42.

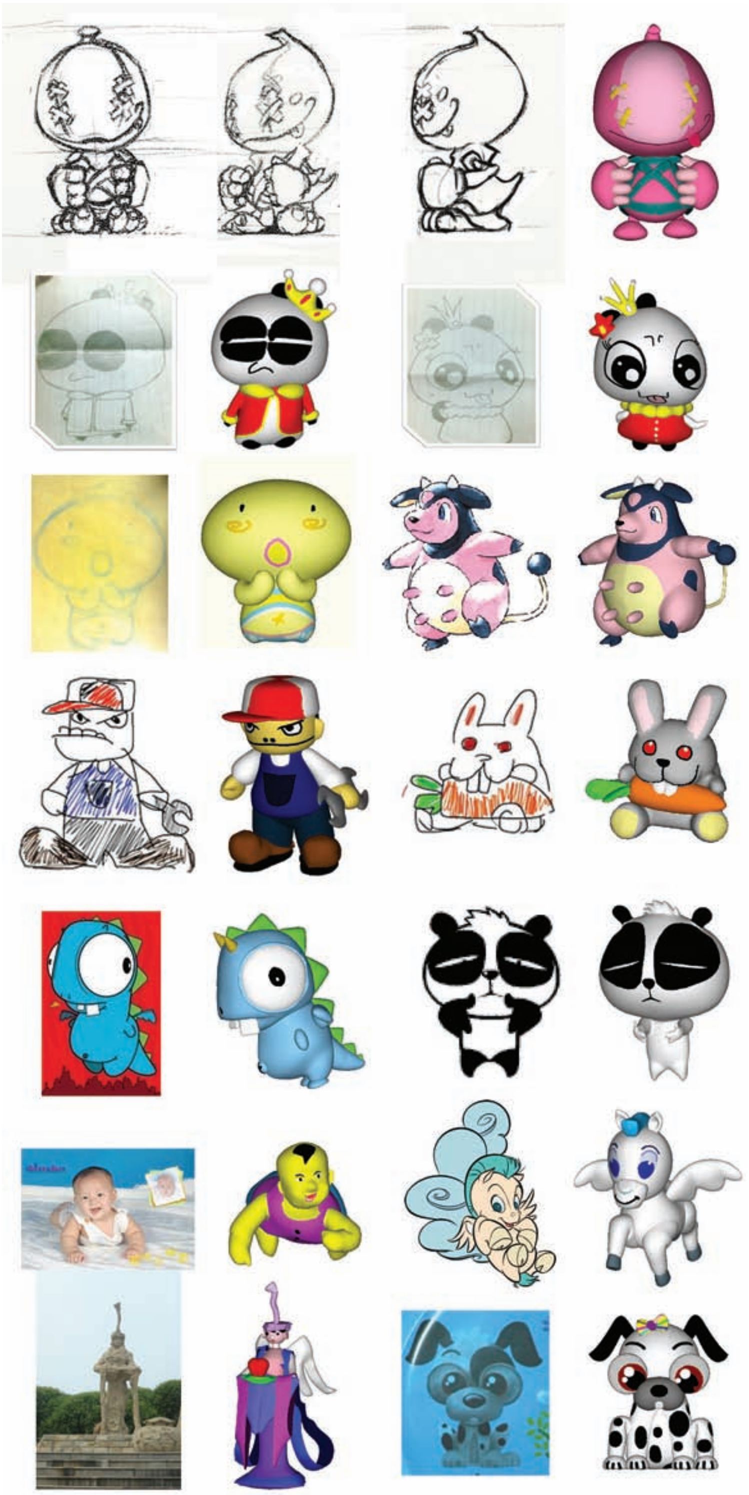

Figure 10. The 2D reference drawings and final toy models designed by novices in a university general-elective course. The novices liked modeling by editable sketches very much.

Yong-Jin Liu *is an associate professor in Tsinghua University's Department of Computer Science and Technology. His research interests include computer graphics and computer-aided design. Liu has a PhD in mechanical engineering from the Hong Kong University of Science and Technology. Contact him at liuyongjin@tsinghua.edu.cn.*

Cui-Xia Ma *is an associate professor at the Institute of Software, Chinese Academy of Sciences. Her research interests include human-computer interaction and computer-aided design. Ma has a PhD in computer science from the Institute of Software, Chinese Academy of Sciences. Contact her at cuixia@ios.cn.*

Dong-Liang Zhang *is a senior researcher at Liveforce. His research interests are 3D modeling, simulation, and computer-aided design for toys and garments. Zhang has a PhD in mechanical engineering from the Hong Kong University of Science and Technology. Contact him at livesforce@yahoo.com.*