DOI:10.1145/1785414.1785436

Interactive computer graphics would rival word-processing and presentation programs for everyday communications.

BY TAKEO IGARASHI

Computer Graphics for All

COMPUTER GRAPHICS IS a commodity. Sophisticated computer-generated imagery is everywhere—feature films, TV programs, video games, even cellphones—but most of it is created by professionals. Few people actually create computer graphics in their daily lives because most authoring tools are designed for professionals or dedicated amateurs following intensive training. This is unfortunate, because computer graphics could be a powerful communication tool for everyone.

Consider desktop publishing. Centuries ago, only a small number of professionals worked in the printing industry. When computer-based printing emerged as an alternative in the late 20th century, it, too, was initially limited to professionals. However, the widespread use of personal computers and easy-to-use graphical user interfaces quickly made high-quality printing accessible to the general public. Today, just about everyone uses word processors on a daily basis to create documents that communicate ideas to friends and colleagues. Computer graphics has not yet achieved such mass-market appeal.

Unlike with traditional physical media, consumers today create electronic content to share through the Internet. Most media are still text-based, as with email, blogs, and Twitter, but more and more include images, videos, animations, and other multimedia con-

» key insights

- **Computer-graphics authoring should be accessible to the general public.**
- **Designing these systems starts with what is natural to humans rather than what is natural to a computer.**
- **Most traditional research focuses on experts' high-end use of technology; here, our main target is the casual use of technology by nonprofessionals.**

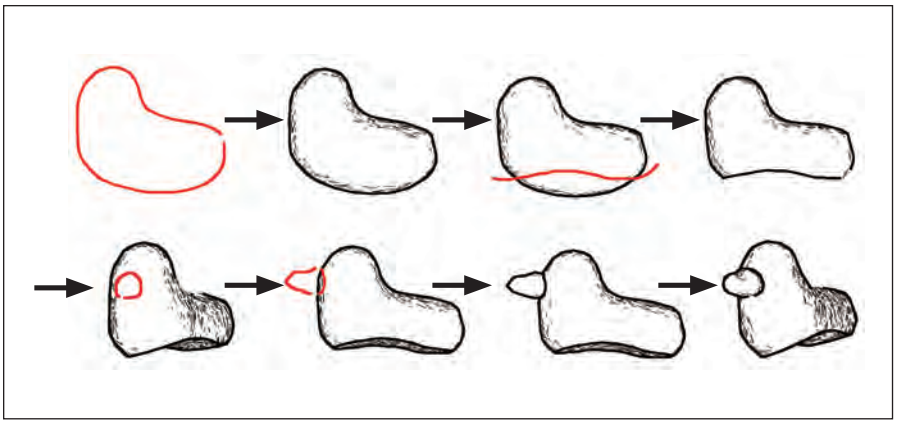

Figure 1. Modeling session in Teddy. Users create 3D models using simple sketching operations.

Figure 2. Screenshot of Teddy and sample 3D models created through the Teddy system.

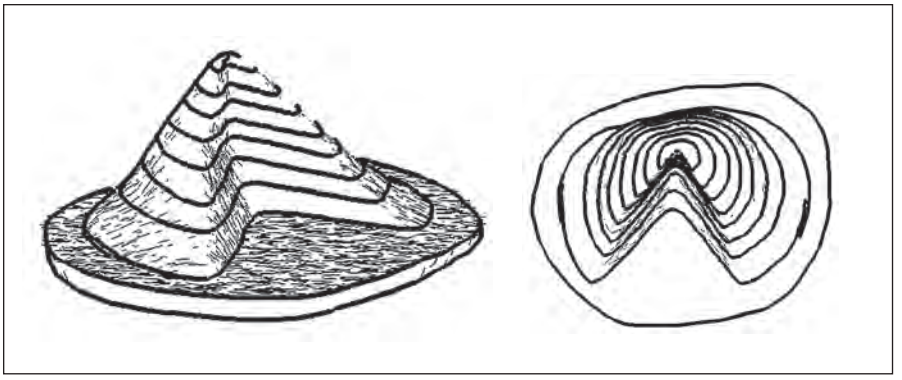

Figure 3. Using Teddy to teach the concept of contour lines.

tent. End users constructing 3D models are also supported by a number of systems, including Google's SketchUp (http://sketchup.google.com/) and modeling tools in *SecondLife* and *Spore*. However, these systems use scaled-down versions of traditional interfaces and still require a certain amount of skill. This article introduces research efforts at the University of Tokyo and Brown University to make computer-graphics authoring accessible to more casual users. To achieve this goal, the author and his collaborators developed easy-to-use prototype systems to create expressive computer graphics more quickly than with traditional interfaces. Examples are sketch-based 3D modeling, clothing manipulation, animation by performance, and 2D shape manipulation. We discuss the user interfaces and technical aspects of these prototype systems, as well as the lessons learned from their development, offering ideas for future research directions.

Most of our work is highly interactive and difficult to explain in written words and still images; please see demonstration videos and prototype systems at http://www-ui.is.s.u-tokyo.ac.jp/~takeo.

Sketching 3D Models

Creating a 3D model in a computer (not necessarily on a screen) is the first step in most 3D computer-graphics applications yet is also the most difficult. Traditional interfaces for 3D modeling programs trace their origins to traditional pencil-and-paper professional drafting. Users place vertices in 3D space by specifying *x*-, *y*-, and *z*-coordinates in a three-view interface, then create polygonal faces (individual polygonal sides of a polyhedron) by connecting these vertices. Alternatively, users start with a simple primitive (such as a sphere or cylinder) and modify it by editing individual vertices and edges. Many editing tools (such as free-form deformation and Boolean operations among solids) are available for designing complicated shapes from simple primitives. Although they might be appropriate for trained professionals designing precise models, they are generally too difficult for first-time users trying to quickly generate meaningful models.

The sketching interface is emerging as an alternative modeling method. Users draw 2D lines on the screen; the system then generates a 3D model automatically, inferring missing depth information. Sketching interfaces for 3D scenes consisting of simple primitives were first introduced in the SKETCH system,[18] allowing users to perform complicated 3D editing operations in a single camera view by combining heuristics. A similar approach is used in commercial systems (such as Google's SketchUp). However, these systems are designed for sketching simple shapes defined by relatively few parameters. Designing them requires specialized training.

Our group at the University of Tokyo developed the Teddy system[11] to address this problem, allowing users to quickly generate interesting 3D freeform models (such as creating a teddy bear by drawing the silhouette of the desired shape) (see Figure 1). The user's strokes are in red; the system infers and draws everything else. The user first draws the silhouette of the base primitive, and the system generates the corresponding 3D geometry. The user then draws a stroke across the model, and the system cuts the model at the line. The user can also add parts to the base model by drawing two strokes; Figure 2 shows several 3D models created this way.

We do not expect Teddy to replace traditional 3D modeling tools. Rather, it will create new 3D modeling applications that are useful to nonexperts, including children, who want to play with 3D graphics for fun. Introduced at the SIGGRAPH conference in 1999, Teddy is used in several current commercial video games to permit players to create their own characters. Using it is a useful way for experts to express their ideas quickly in early design phases. A commercial 3D modeling package, Shade (available only in Japan, http://shade.e-frontier.co.jp/), includes an extension to Teddy as a plug-in for generating rough sketches. Finally, and most important, Teddy is useful for communicating 3D concepts face to face. In a classroom, for example, a teacher could quickly draw a model of bacteria, showing its cross section to explain its internal structure. In a hospital, a medical doctor could draw a model of a stomach to help explain a patient's stomach disease.

In 2003, to test the idea, we conducted a trial in a high school geography class in Chiba, Japan. Teaching 3D concepts (such as mountains and valleys), a geography teacher would have difficulty explaining them using traditional 2D media like a blackboard. Sketching in 3D can help address this problem. A convincing example is the teaching of contour lines using the Teddy system (see Figure 3) in which the teacher first shows a 3D model of a mountain, then draws several horizontal lines in the side view, saying the lines indicate equal height intervals. The teacher then changes the viewpoint to show the mountain and the lines from the top. This way, students understand the relationship between the closed lines on the map (contour lines) and the 3D geography, not just mountains, ridges, and valleys.

Clothing Manipulation

3D characters must also be dressed properly. The computer-graphics research community has actively investigated the physical simulation of cloth, today producing realistic cloth simulations. However, the initial simulated-cloth configuration must be set manually, and the user interface for manipulating cloth is primitive. A typical approach is to place rigid cloth patches around the target body, combining 3D translation and rotation before starting the simulation—a tedious process. Moreover, users have difficulty changing the way the garment is worn once they've placed it on a character. Standard systems allow users to freely move individual vertices through direct manipulation, but it causes a large local distortion (stretching), making it difficult to achieve global movement.

In 2001, our group at Brown University developed clothing-manipulation techniques to address these issues.[10] To put a garment on a character, users first draw free-form marks on both the garment and the character to indicate positional correspondence (see Figure 4).

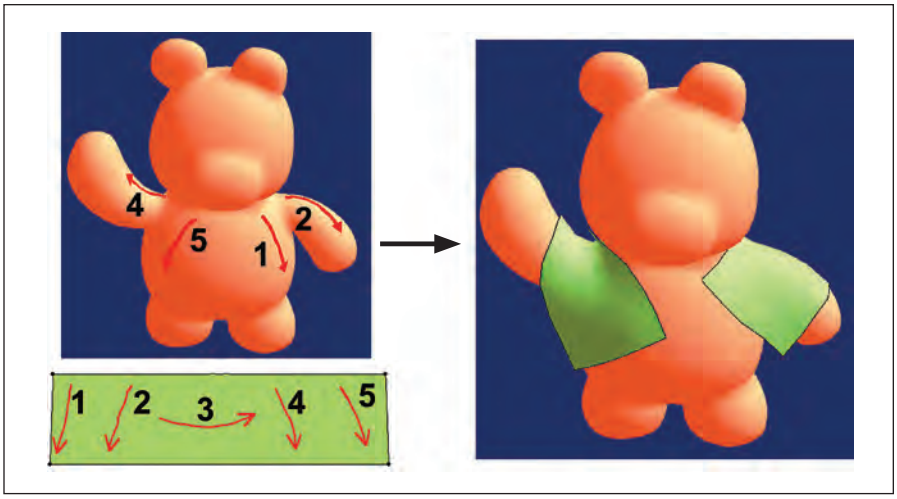

Figure 4. Users draw marks on the character and cloth; the system then places the cloth on the character.

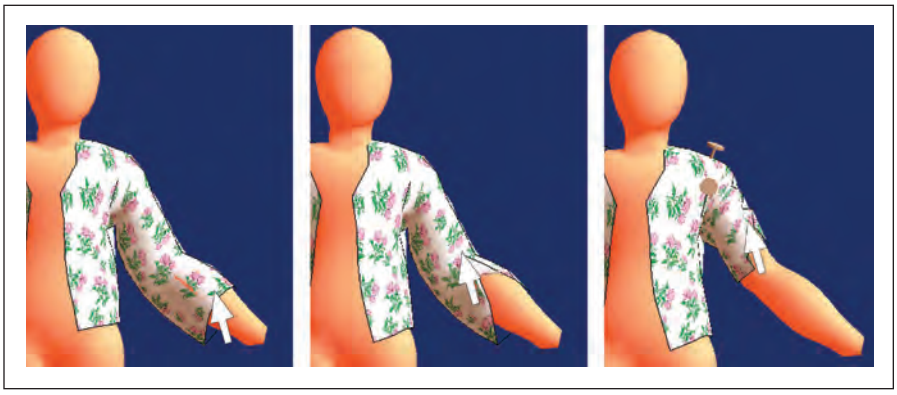

Figure 5. Dragging the cloth onto the character: left, before dragging; center, the result of traditional vertex dragging; right, the result of our clothing-manipulation method.

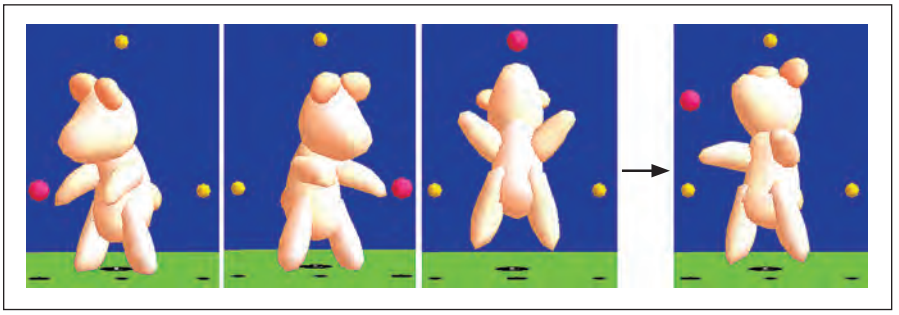

Figure 6. Spatial keyframing. Users specify three key poses (left), then freely control the character by dragging the red ball (right).

The system then places the garment on the character so the marks on the garment match the corresponding marks on the character. The system uses a simple relaxation process during placement to prevent stretching and squashing, even if the lengths of the corresponding marks are different. Working with only a few strokes, users are able to place reasonably complicated garments on any character.

Once a garment is on a character, users can grab any point of the garment and drag it onto its surface (see Figure 5). Unlike standard vertex dragging, in which a single vertex is moved while relying on subsequent simulation to move other vertices, this dragging operation moves all vertices of the cloth mesh directly, causing global movement. To achieve global movement, the movement vector of the dragged vertex is propagated to the complete cloth mesh along the surface of the character.

This technique allows even novice users to quickly test many different ways of dressing virtual characters. We also expect it to be useful for designing real garments as well. The cloth representation and simulation are limited in the prototype system implemented in 2001, but the basic user interface should still be applicable to today's more sophisticated cloth representation.

The technical contribution of this work is the behavior of the cloth material in response to user input. It not only follows physical principles (such as gravity and collision) but behaves proactively to assist a user's design process; for example, the cloth automatically unfolds local folds based on the assumption that users do not want to see accidental local folds unless they explicitly require them. Such built-in intelligent behavior of passive materials can be useful in other domains; we are now testing it in knot- and hairstyle-design systems.

Performance-Driven 3D Animation

Keyframing is the most popular method for designing character animation. The user specifies the pose of the character at each time point, and the system interpolates the key poses at runtime. Though many other methods (such as motion capture and procedural animation) are available, keyframing is by far the most popular approach due to its simplicity and versatility. But manually defining so many keyframes is tedious. Moreover, novice users experience great difficulty designing natural-looking motion through discrete sets of poses. The result tends to look mechanical while lacking the rich textures seen in the motion of living things.

A live demonstration is the simplest approach to designing motion, in which a user moves the target character in real time and the system records the motion, like dancing a teddy bear in front of a video camera. However, moving a character with many joints is difficult when using a standard input device like a mouse. Though possible to demonstrate the motion of each joint one at a time,[6] synchronizing individual motions is difficult.

Our group at Brown University developed a spatial keyframing method to address this problem.[9] With it, the user first sets a group of key poses in the 3D or 2D space; a pose is associated with a position in a space. The user then moves the cursor in that space, and the system sets the character pose by blending the key poses around the cursor position (see Figure 6). The user is thus able to design interesting whole-body character motion (such as juggling and dancing) by recording simple cursor movements.

This technique is well suited to defining expressive motion (such as to show joy or sadness). The resulting motion is much more alive than motion generated through traditional keyframing because the motion directly mirrors the operator's natural hand movement. However, motion dominated by physical factors (such as jumping and running) is better supported by physical approaches.[7]

We expect spatial keyframes to be a useful intermediate representation for 3D characters. Current 3D character representation consists of geometry, texture, rigs, and possibly predefined motions. Users who want to define a new motion must specify individual poses one at a time. Specialized tools include Maya's set-driven keys (http://caad.arch.ethz.ch/info/maya/manual/UserGuide/Animation/KeyframeMoPath/03_understanding_key.doc5.html) and the Waldo input device (http://www.character-shop.com/waldo.html), though neither is designed for the blending of key poses. By providing predefined spatial keyframes (a set of natural poses) for a character, users can create new motion very quickly by moving the control cursor. This can make it much easier for inexperienced users to make characters move at will.

2D Shape Manipulation

In the physical world, one can hold an object (such as a teddy bear) with two hands and freely manipulate it through rotating, stretching, squashing, and bending motions. Standard 2D drawing programs provide poor support for such shape manipulation, allowing only simple editing operations (such as scaling and rotation). Not only do these operations require a complicated combination of tools, the result for the user simply doesn't feel like manipulating a physical entity.

In 2004, our group at the University of Tokyo developed a novel manipulation technique to address this problem.[8] Users are thus able to select arbitrary points as handles on a 2D shape, then freely manipulate the shape by moving the handles (see Figure 7). They can relocate the shape by setting a single handle to rotate, stretch, and squash the shape. Users also swing a head or stretch an arm by setting handles on the corresponding positions. The

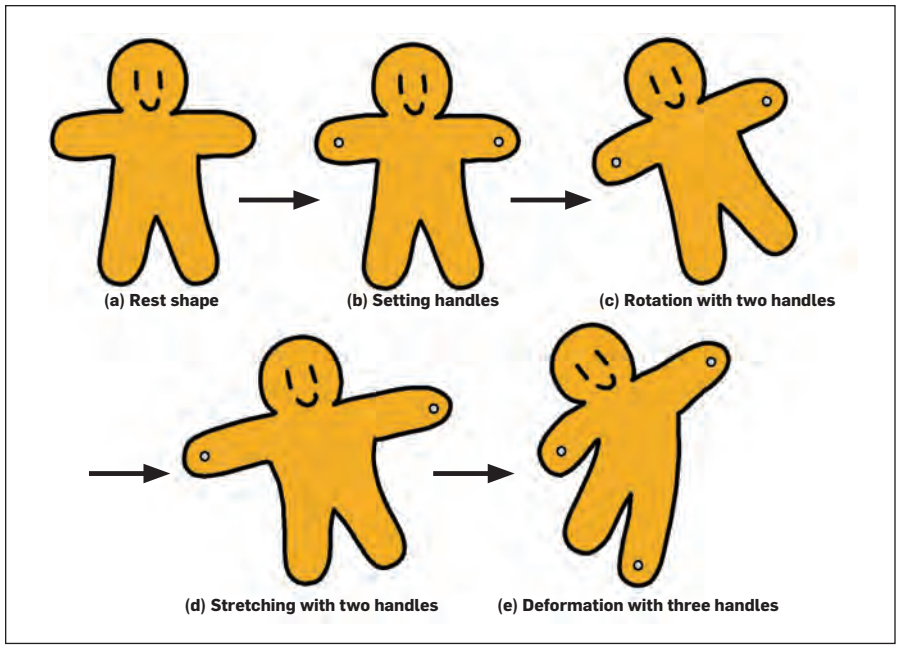

Figure 7. As-rigid-as-possible shape manipulation. Users place handles on the drawing, then manipulate it by moving the handles.

shape deforms naturally in response to user input; for users it feels like they're manipulating a physical object.

Traditional computer-based methods for shape manipulation are roughly divided into three categories:

Skeleton.[13] The user embeds a skeletal structure inside the shape and controls it to deform the shape. However, embedding a skeleton in each shape is tedious, and the approach does not work for stretching and squashing;

Spatial deformation.[14] The user defines a spatial mapping using several control points, then deforms the shape according to the mapping function. However, mapping functions do not consider the rigidity of the shape and result in unnatural deformation; and

Physics-based. This approach simulates the deformation process of physical material.[12] However, the computation is not fast or stable enough to provide real-time feedback to a global deformation caused by user operations.

Our method takes a completely geometric approach, defining an energy function that measures the amount of geometric deformation, then minimizes it using an optimization method. We designed the energy to give a closed-form solution to the problem. In it, the system obtains the deformation by solving two large sparse linear-matrix equations in sequence, a very fast and perfectly stable approach.

It is also particularly useful for creating 2D animations. Traditional animation artists assemble many slightly different drawings to create an animation. In our shape-manipulation system MovingSketch (http://www-ui.is.s.u-tokyo.ac.jp/~takeo/research/rigid/movingsketch/index.html), users create an interesting animation by drawing a character and recording the manipulation process. Using a multi-touch input device,[16] they grab a character with both hands and manipulate it to create an animation (see Figure 8). We tested the system with children in an educational TV program in Japan and found that even elementary-school students could quickly generate reasonably interesting animations.

Lessons Learned

Each of these projects addresses a specific problem, with technical contributions being rather independent. However, emerging from them are common guidelines for designing a compelling user experience:

Natural to humans. First, start with what is natural to a human rather than with what is natural to a computer. The computer represents a 3D model with a collection of 3D points and their connections; traditional computer-aided-design systems ask users to provide this information directly. Advanced systems represent a model with a sequence of editing operations, but most of them still require that users be aware of points and faces. Similarly, a computer represents a 2D drawing with its position and orientation. Traditional drawing systems ask users to directly control these parameters; that is, traditional systems expose the underlying representation to the user directly. Though it is the most straightforward way to implement a system, the result means difficulty for novice users. That's why we start by identifying the most natural operations for a human referring to real-world examples, then developing an algorithm that maps

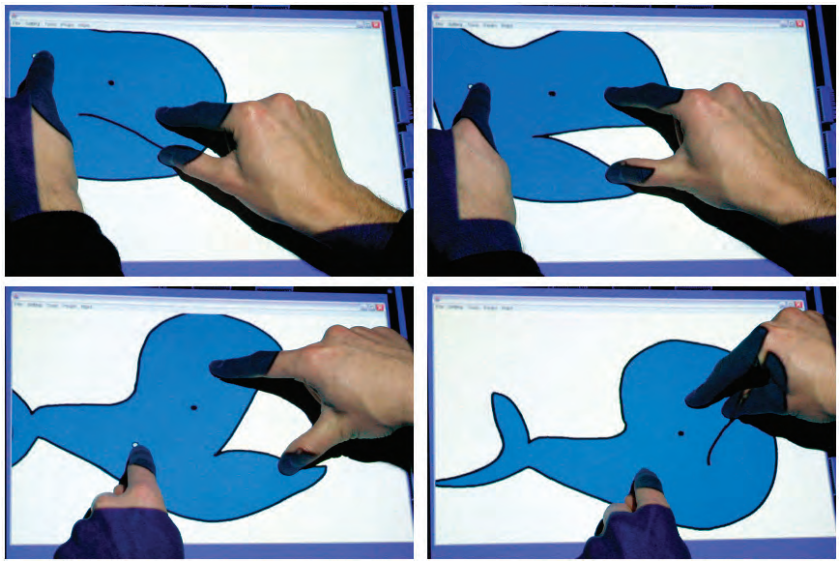

Figure 8. Bimanual manipulation of a drawing.

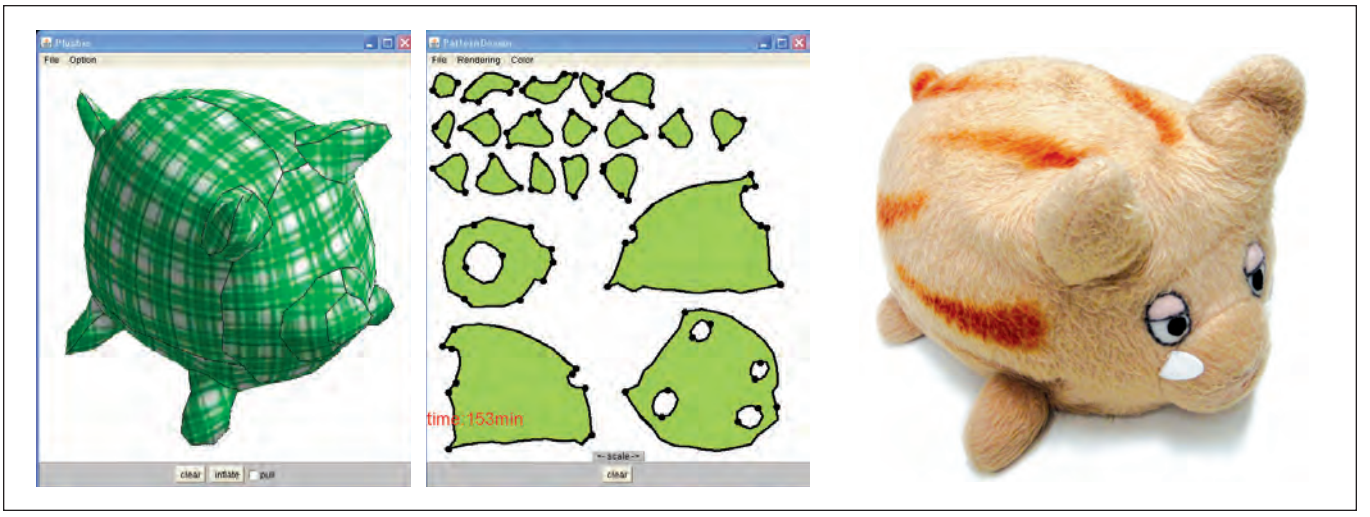

Figure 9. Screenshot of the Plushie system and plush toy designed with the system.

them to computer representations. In the case of 3D modeling, we learned from sketching activity on real paper. For 2D animation, we learned from children playing with a real plush toy using both hands.

Instant feedback. Instant feedback is critical to real-time interaction. It allows for graceful learning through casual trial and error while supporting creative exploration through rapid experimentation. To provide a rich and comfortable user experience, three opportunities for executing computation should be used: One is computation during mouse dragging, a computation that must be very fast (on the order of 0.1 seconds). The second is computation right after a mouse click or dragging; it can be somewhat slower (about a full second). And the third is running a computation in the background while the user is looking at a result. The sketching interface is effective because it gives a system the opportunity to execute a heavyweight computation right after the completion of a sketch (mouse release) that would otherwise be too time-consuming during a mouse drag. In 2D animation, the system computes time-consuming matrix factorization when a pin is added or removed, applying fast back-substitution during dragging. The clothing-manipulation system exploits idle time to refine the cloth configuration.

Right target task. System designers must choose and focus on the right target task to achieve the first two goals. Developers try to address a range of tasks, overloading the interface with too many functions, as in professional systems like Maya. In theory, including more functions could expand the range of user options but also require intensive training and reduce what casual users are able to do in the system. Carefully limiting functional scope, designers provide an optimized interface and algorithm for the task in exchange for losing some rarely used functions. Teddy is designed for rotund models (such as stuffed animals), freeing users from having to specify depth information each time. The clothing-manipulation system simplifies the interface and accelerates the computation by focusing on the cloth on the surface of the body. System designers are better off tapping user creativity than constraining it with many predefined functions. A simple, well-designed interface allows users to apply their imaginations to complete tasks beyond the system designer's original assumptions, as in terrain sketching with Teddy.

We would also like to share some general lessons learned after the original publication of these research results in 1999.[11] First, even though a sketching interface does lower the threshold, 3D modeling remains difficult. The main difficulty is control of 3D rotation with a 2D input device. We observed that many test users failed to rotate a model to the desired orientation. It is therefore desirable to give users rotation-free modeling methods or a significantly easier rotation interface. Second, though we tried to extend these techniques to support more advanced modeling operations, we were unsuccessful. The more operations we tried to support, the more complicated the interface became, and the original advantages disappeared. We therefore explored new application domains instead of focusing on the same problem. Finally, the tools we've outlined here were generally better received by users with no prior experience in 3D modeling or animation. Users who had previously worked with computer graphics had their own preferred tools and did not show much interest in Teddy. Those without prior experience saw great potential. We encourage researchers working on similar problems to not be intimidated by negative reactions from existing users but to try finding new users outside the existing user community. This will ultimately expand application of computer graphics.

Future Directions

In addition to improving the tools discussed here, we plan to work on other aspects of computer-graphics authoring in the future, including two notable problems:

Designing interactive behaviors. Interactivity is an important aspect of computer-generated media. Not only do users passively watch predefined imagery, they also interact with computer imagery (such as by poking a character) to observe its response. The systems we've introduced here are all interactive as authoring tools, but the content they produce is noninteractive; 3D models and 2D animation created this way do not respond to user input. End-user design of interactive behavior is an exciting but challenging research direction.

Several earlier research efforts sought to achieve end-user design of interactive behavior. One involved making traditional programming (scripting) accessible to casual users through a highly visual editing environment.[3] In the system, users write a program using simple drag-and-drop operations without making syntax errors. Another involved using programming by demonstration[5] for character animation[17]; in it, users demonstrate the desired interactive behavior of a character, with the system learning the pattern from the demonstration. Programming with visual replacement rules is a promising approach for defining a character's interactive behavior.[4] The user specifies before-and-after pairs; at runtime, the system compares the scene configuration with the *before* patterns, replacing it with *after* patterns when the match is identified.

Though these experiments produced interesting initial results, designing the arbitrary interactive behavior of a virtual agent is often prohibitively difficult. We are particularly interested in teaching interactive behavior to physical agents (robots). End-user programming for robot behavior has been tested in some systems[1] but is still limited to basic motions. Programming by demonstration for robots has also been reported but is used mainly for acquiring physical-manipulation skills.[2] Techniques developed in the user-interface-research community that should be applicable to human–robot interaction represent an interesting research direction.

Designing real-world objects. The systems outlined here were all designed for virtual representations; one can produce interesting graphics on the computer screen but cannot touch or use them in the real world. Then there's development of end-user tools for designing physical objects (such as furniture and clothing). The idea is to help people custom-design the things they will use instead of having to buy manufactured products in stores. Objects designed by users themselves should satisfy their needs more directly and produce greater satisfaction.

Unlike professional designers, the typical consumer generally lacks the professional knowledge needed to design physical objects. Inexperienced consumers could easily create a bag that is not sturdy enough or a chair that cannot stand up. One promising approach is to introduce physics into the modeling process. Traditional modeling systems ignore physics, possibly producing physically inappropriate results, as in, say, objects that penetrate one another. It might be possible to help users avoid these issues by considering physical principles within a modeling system.

In 2006, our first such experiment involved a design system for plush toys.[15] Users would interactively draw a sketch on the screen, and the system would then automatically generate a 3D plush toy model, as in the Teddy system. In addition, the system simultaneously generated a 2D cloth pattern corresponding to the 3D geometry, allowing the user to create a physical plush toy by cutting the cloth according to the generated pattern (see Figure 9). Internally, the system first generated a 2D cloth pattern, then ran a simple physical simulation to predict the 3D shape of the resulting toy. This way, even young children would be able to design their own plush toys just by sketching.

The idea of 3D modeling with physical simulation is very powerful. We expect future modeling systems to consider various physical constraints in the background (such as collisions and stability), freeing users from low-level physical concerns and allowing them to concentrate on more important high-level design concerns. We plan to test this idea in a number of target domains, including furniture and clothing design.

Conclusion

This article introduced our efforts to make computer-graphics authoring accessible to the general public, making it as much a daily communication tool as word processing and presentation applications. What most defines our research is its focus on end users. This opens up new application possibilities for existing technologies while posing unique technological challenges for interface researchers and developers. We look forward to more computer-science researchers participating in this fertile field.

Acknowledgments

I would like to thank Satoshi Matsuoka, Hidehiko Tanaka, John F. Hughes, Tomer Moscovich, Yuki Igarashi, Maneesh Agrawala, and Masahiko Inami for their contributions and comments. C

References

1. Baum, D. and Zurche, R. *Definitive Guide to Lego Mindstorms.* Apress, New York, 2000.
2. Billard, A., Calinon, S., Dillmann, R., and Schaal, S. Robot programming by demonstration. In *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds.. Springer, New York, 2008, 1371–1394.
3. Cooper, S., Dann, W., and Pausch, R. Teaching objects first in introductory computer science. In *Proceedings of the ACM Technical Symposium on Computer Science Education* (Reno, NV, Feb. 19–22). ACM Press, New York, 2003, 191–195.
4. Cypher, A. and Smith, D.C. KidSim: End-user programming of simulations. In *Proceedings of the ACM Conference on Computer Human Interaction* (Denver, May 7–11). ACM Press, New York, 1995, 27–34.
5. Cypher, A. *Watch What I Do: Programming by Demonstration.* MIT Press, Cambridge, MA, 1993.
6. Dontcheva, M., Yngve, G., and Popović, Z. Layered acting for character animation. In *Proceedings of ACM SIGGRAPH* (San Diego, CA, July 27–31). ACM Press, New York, 2003, 409–416.
7. Fang, A.C. and Pollard, N.S. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3 (July 2003), 417–426.
8. Igarashi, T., Moscovich, T., and Hughes, J.F. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics 24*, 3 (July 2005), 1134–1141.
9. Igarashi, T., Moscovich, T., and Hughes, J.F. Spatial keyframing for performance-driven animation, In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, July 29–31). ACM Press, New York, 2005, 107–115.
10. Igarashi, T. and Hughes, J.F. Clothing manipulation. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (Paris, Oct. 27–30). ACM Press, New York, 2002, 91–100.
11. Igarashi, T., Matsuoka, S., and Tanaka, H. Teddy: A sketching interface for 3D freeform design. In *Proceedings of ACM SIGGRAPH* (Los Angeles, Aug. 8–13). ACM Press, New York, 1999, 409–416.
12. James, D.L. and Pai, D.K. ArtDefo: Accurate real-time deformable objects. In *Proceedings of ACM SIGGRAPH* (Los Angeles, Aug. 8–13). ACM Press, New York, 1999, 65–72.
13. Lewis, J.P., Cordner, M., and Fong, N. Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH* (New Orleans, July 23–28). ACM Press, New York, 2000, 165–172.
14. MacCracken, R. and Joy, K.I. Freeform deformations with lattices of arbitrary topology. In *Proceedings of ACM SIGGRAPH* (New Orleans, Aug. 4–9). ACM Press, New York, 1996, 181–188.
15. Mori, Y. and Igarashi, T. Plushie: An interactive design system for plush toys. *ACM Transactions on Graphics 26*, 3 (July 2007).
16. Rekimoto, J. SmartSkin: An infrastructure for freehand manipulations on interactive surfaces. In *Proceedings of ACM Conference on Human Computer Interaction* (Minneapolis, Apr. 20–25). ACM Press, New York, 2002, 113–120.
17. Young, J.E., Igarashi, T., and Sharlin, E. Puppet Master: Designing reactive character behavior by demonstration. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation* (Dublin, July 7–9). ACM Press, New York, 2008, 183–191.
18. Zeleznik, R.C., Herndon, K.P., and Hughes, J.F. SKETCH: An interface for sketching 3D scenes. In *Proceedings of ACM SIGGRAPH* (New Orleans, Aug. 4–9). ACM Press, New York, 1996, 163–170.

Takeo Igarashi (takeo@acm.org) is an associate professor in the Department of Computer Science in the Graduate School of Information Science and Technology at The University of Tokyo.