Freeform User Interfaces for Graphical Computing

Takeo Igarashi

Department of Computer Science, The University of Tokyo / PRESTO, JST
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Tokyo, JAPAN
takeo@acm.org
http://www-ui.is.s.u-tokyo.ac.jp/~takeo

Abstract. It is difficult to communicate graphical ideas or images to computers using current WIMP-style GUI. Freeform User Interfaces is an interface design framework that leverages the power of freeform strokes to achieve fluent interaction between users and computers in performing graphical tasks. Users express their graphical ideas as freeform strokes using pen-based systems, and the computer takes appropriate actions based on the perceptual features of the strokes. The results are displayed in an informal manner to facilitate exploratory thinking. This paper explores the concept of Freeform UI and shows its possibilities with four example systems: beautification and prediction for 2D geometric drawing, a stroke-based 3D navigation, an electronic office whiteboard, and a sketch-based 3D freeform modeling. While Freeform UI is not suitable for precise, production-oriented applications because of its ambiguity and imprecision, it does provide a natural, highly interactive computing environment for pre-productive, exploratory activities in various graphical applications.

1 Introduction

Graphical User Interface (GUI) has been the predominant user interface paradigm for almost 30 years. But because the purpose of computing is changing, we clearly need next-generation user interface framework. In the near future, computers' main application will no longer be as a tool for supporting knowledge workers in office environments. As they become smaller and still less expensive, they will become ubiquitous and their goal will be to support every aspect of human life. At that stage, a new form of user interfaces, post-WIMP [16] or non-command [13] user interfaces, will be needed. In [13], Nielsen argued that current GUI is essentially the same as command-line user interface in that users have to translate their tasks into machine-understandable sequences of commands. Pressing buttons or selecting items in menus in GUI is essentially identical to typing commands in command-line user interface. In non-command user interfaces, computers take appropriate action based on the users activity, allowing the user to concentrate on the task itself without worrying about commands.

Candidates for post-WIMP, non-command user interface include virtual realities and augmented realities, multi-modal and multi-media interfaces, natural language interfaces, sound and speech recognition, portable and ubiquitous computers. Each

new interface is designed to support specific new uses of computers. The increasing number of applications dealing with three-dimensional information require virtual reality techniques and various three-dimensional input devices. The need to support people in situations where one cannot use hands or keyboards has spurred the growth of voice input technologies. Highly complicated, spatial applications gave birth to the idea of physical (graspable or tangible) interfaces that can provide more affordable, space-multiplexed input channels. The essence of the next-generation user interface is its diversity. While current user interfaces are characterized simply as "WIMP-style GUI," post-WIMP or non-command user interfaces will be characterized as collections of task-oriented, tailored interfaces. An important task for user interface research is to identify an emerging application domain and find the ideal user interface for that domain beyond WIMP-style GUI.

This paper explores a user interface framework, Freeform User Interfaces, as a post-WIMP, non-command user interface in the domain of graphical interaction. Current point-click-drag style interaction is suitable for specific kinds of graphical interaction, namely object-oriented graphics such as block diagrams or flow charts. However, the point-click-drag interface does not work well for expressing arbitrary graphical ideas or geometric shapes in computers. The user has to do this manually by placing many control points one by one or combining editing commands in a nested menu. On the other hand, people have been using pen and paper to express graphical ideas for centuries. Drawing freeform strokes is a convenient, efficient, and familiar way to express graphical ideas. Freeform UI is an attempt to bring the power of freeform strokes to computing.

Section 2 introduces the concept of Freeform UI, a pen-based non-command user interface for graphical applications. We define the concept with three properties: stroke-based input, perceptual processing, and informal presentation. Section 3 briefly introduces four independent example systems embodying the idea of Freeform UI. They as a whole form a concrete basis for discussing the nature of Freeform UI. Section 4 discusses the limitation of Freeform UI and several design principles to mitigate the problems.

2 Freeform User Interfaces

Freeform UI is an interface design framework using pen-based input for computer-supported activities in graphical domains. In Freeform UI, the user expresses visual ideas or messages as freeform strokes on pen-based systems, and the computer takes appropriate action by analyzing the perceptual features of the strokes. This is based on the observation that freeform sketching is the most intuitive, easiest way to express visual ideas. The fluent, lightweight nature of freeform sketching makes Freeform UI suitable for exploratory, creative design activities. Freeform UI embodies a non-command user interface for two- and three-dimensional graphical applications in that the user can transfer visual ideas into target computers without converting the ideas into a sequence of tedious command operations.

Specifically, Freeform UI is characterized by the following three basic properties: the use of pen-based stroking as input, perceptual processing of strokes, and informal presentation of the result. We describe each property in detail in the following subsections.

2.1 Stroke-based Input

Freeform UI is characterized by its use of strokes as user input. A stroke is a single path specified by the movement of a pen and is represented as a sequence of points internally. Stroking is usually recognized as a dragging operation in a standard programming environment: it is initiated by "button press" event, followed by a sequence of "mouse move" event, and terminated by "button release" event. However, stroking is actually a significantly different interface model than dragging. In short, stroking corresponds to physical drawing activity using real pen and paper, while dragging corresponds to a physical grab-and-move operation of objects. During a stroking operation, the trajectory of the pen's movement is shown on the screen, and the system responds to the event when the user stops stroking by lifting the pen. The system's reaction is based on the entire trajectory of the pen's movement during the stroking, not just the pen's position at the end (Fig. 1). In contrast, in a typical dragging operation, the current cursor position is shown on the screen. Possibly, the object shape specified by the current cursor position is shown as a feedback object, but the trajectory of the cursor movement is not shown. The system's action is based on the final cursor position and possibly the starting position of dragging. In stroking, the user first imagines the desired stroke shape and then draws the shape on the screen at once, while the user constantly adjusts the cursor position observing the feedback objects during dragging.

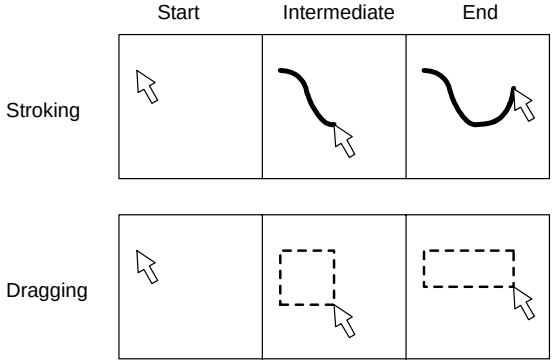

Fig. 1. Stroking vs. dragging.

Pen-based stroking is an intuitive, fast, and efficient way to express arbitrary graphical ideas in computing environments. This is because a pen-based stroking operation, or sketching, has been for centuries the primary interaction technique for expressing graphical ideas, and is therefore familiar to us. Specifically, stroking is suitable for quickly entering rough images that internally consist of many parameters

from the computer's point of view. On the other hand, mouse-based dragging is suitable for more-delicate control of simple parameters. Dragging has been the dominant interaction technique because traditional computer-based drawing applications are designed for the careful construction of precise diagrams. The argument of this paper is that graphical computing in the future should support informal drawing activities and thus require a pen-based stroking interface.

2.2 Perceptual Processing

The next important property that characterizes Freeform UI as a non-command user interface, and that makes Freeform UI different from plain pen-based scribbling systems, is its advanced processing of freeform strokes inspired by human perception. Scribbling programs such as those used in commercial electronic whiteboards simply convert the user's pen movement into a painted stroke on the screen without any further processing. Character-recognition and gesture-recognition systems convert a stroke into a predefined character or command, using pattern-matching algorithms. In these recognition systems, the output of the recognition is represented as a single symbol. The stroking operation in these systems is essentially equivalent to key-typing and button-pressing. "Perceptual processing" refers to mechanisms that infer information from simple strokes that is richer than mere symbols. The idea behind perceptual processing is inspired by the observation that human beings perceive rich information in simple drawings, such as possible geometric relations among line primitives, three-dimensional shapes from two-dimensional silhouettes (Fig. 2). Perceptual processing is an attempt to simulate human perception at least in limited domains.

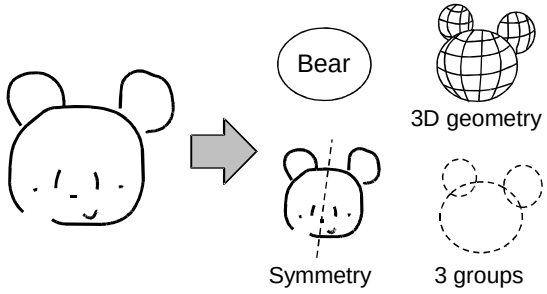

Fig. 2. Human beings perceive rich information in a simple drawing.

The goal of perceptual processing is to allow the user to perform complicated tasks with a minimum amount of explicit control. In traditional command-based interfaces, the user must decompose a task into a sequence of machine-understandable, fine-grained command operations, then input the commands one by one. As we discussed in Section 1, non-command user interfaces try to avoid this process and allow the user to directly interact with tasks without worrying about low-level commands. Freeform UI frees users from detailed command operations by this perceptual processing of freeform strokes. For example, Pegasus (Section 3.1) frees the user from tedious geometric operations such as rotation and duplication by automatically inferring desired

geometric constraints, and Teddy (Section 3.4) eliminates the manual positioning of many vertices in 3D space by automatically constructing 3D geometry from the input stroke. This simplicity also significantly reduces the effort spent on learning commands. In traditional command-based systems, the user has to learn many fine-grained editing commands to do something simple. In Freeform UI, on the other hand, the user can do a variety of things simply after learning a single operation.

2.3 Informal Presentation

The last property of Freeform UI is informal presentation of contents. The system displays the materials to manipulate or the result of computation in an informal manner, using sketchy representation without standard, cleaned-up graphics. This informal presentation is important not only for an aesthetically pleasing appearance, but also to arouse appropriate expectations in the user's mind about the system's functionality. If the system gives feedback in precise, detailed graphics, the user naturally expects that the result of computation will be precise and detailed. In contrast, if the system's feedback is in informal presentation, the user can concentrate on the general structure of the information without worrying about the details too much. The importance of informal presentation in exploratory design activities has been discussed in many papers [1,2,13,17].

Several experimental systems implemented sketchy presentation techniques. Strothotte et al. introduced a non-photorealistic renderer for an architectural CAD system [15]. The system used irregular curves for representing straight line segments to make them appear hand-drawn. The SKETCH system [18] also used non-photorealistic rendering to give a sketchy appearance to a 3D scene being constructed. The system intentionally displaced the vertex position when rendering projected 2D line segments. Teddy uses a real-time pen-and-ink rendering technique developed by Markosian et al. [10]. It efficiently detects the silhouette lines of a 3D model, and renders the silhouettes in various styles.

While these systems are designed for 3D graphics, some systems introduced sketchy rendering for 2D applications. The EtchaPad system [11] used synthesized wiggly lines for displaying GUI widgets in order to give them an informal look. Other systems employ the user's freeform strokes as-is to represent recognized primitives without cleaning up the drawings. SILK [9] allows the user to interact with the GUI widgets sketched on the screen. The Electronic Cocktail Napkin system [3] also retains and displays the as-inked representation of hand-drawn graphical primitives. Pegasus used intentionally thick line segments to show beautified drawings to give them an informal look.

3 Example Systems

This section presents four independent example systems embodying the idea of Freeform UI. While each of these systems contributes independently to the improvement of existing applications, taken as a whole they form a concrete basis for discussing the nature of Freeform UI, including its strengths and limitations.

3.1 Pegasus: Beautification and Prediction for 2D Geometric Drawing [5,6]

Pegasus is a system that allows the user to construct precise illustrations such as shown in Fig. 3 without using complicated editing commands such as copy, flip, and move. The idea is to automate complicated drawing operations by having the computer infer possible geometric constraints and the user's next steps from the user's freeform strokes. Interactive beautification receives the user's free stroke input and beautifies it by considering possible geometric constraints among line segments such as connection, parallelism, and congruence. The system generates multiple alternatives to prevent recognition errors. Predictive drawing predicts the user's next drawing operation based on the spatial relationships among existing segments on the screen.

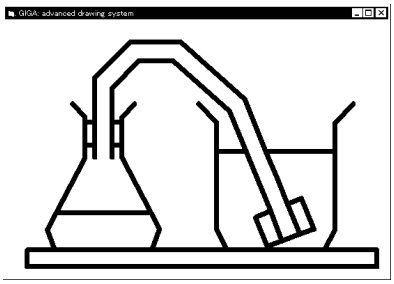

Fig. 3. A diagram drawn using interactive beautification and predictive drawing.

3.2 Path-drawing Technique for Virtual Space Navigation [7]

This technique allows the user to navigate through a virtual 3D space by drawing the intended path directly on the screen. After drawing the path, the avatar and camera automatically move along the path (Fig. 4). The system calculates the path by projecting the stroke drawn on the screen onto the walking surface in the 3D world. Using this technique, with a single stroke the user can specify not only the goal position, but also the route to take and the camera orientation at the goal. This is faster and more intuitive than to turn and advance using arrow buttons or a joystick.

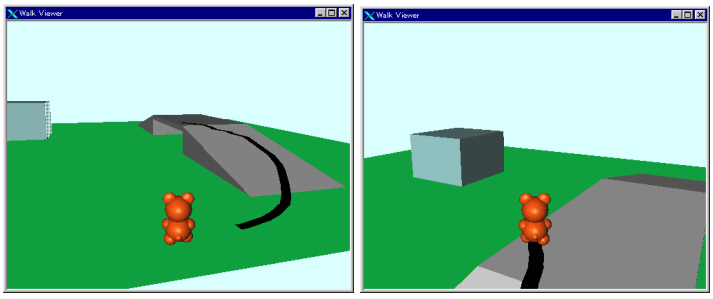

Fig. 4. An example of a path-drawing walkthrough.

3.3 Flatland: An Electronic Office Whiteboard for Informal Activities [4, 12]

Flatland is an augmented whiteboard interface designed for informal office work. Our research has investigated approaches to building an augmented whiteboard in the context of continuous, long-term office use. In particular, the system is characterized by the following three features: techniques for the efficient management of space on the board, the ability to flexibly apply behaviors to support varied domain specific activities, and mechanisms for managing history on the board. We implemented a calculator that takes hand-written numbers as input, map drawing program that takes freeform strokes and turns them into streets and intersections, to-do list manager that organizes handwritten to-do items. These facilities provide support for pre-productive activities, rather than final production work, in an office setting.

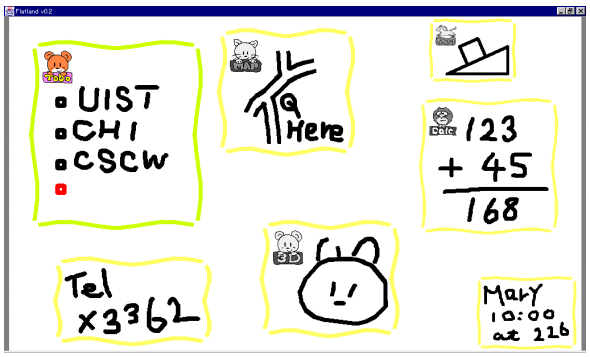

Fig. 5. Flatland example.

3.4 Teddy: A Sketch-based 3D Freeform Modeling System [8]

This technique allows the user to quickly and easily design freeform models, such as stuffed animals and other rotund objects, using freeform strokes. The user draws several 2D freeform strokes interactively on the screen and the system automatically constructs plausible 3D polygonal surfaces. Our system supports several modeling operations, including the operation to construct a 3D polygonal surface from a 2D silhouette drawn by the user: the system inflates the region surrounded by the silhouette, making wide areas fat and narrow areas thin. Teddy, our prototype system, is implemented as a Java program, and the mesh construction is done in real-time on a standard PC. Our informal user study showed that a first-time user typically masters the operations within 10 minutes, and can construct interesting 3D models within minutes.

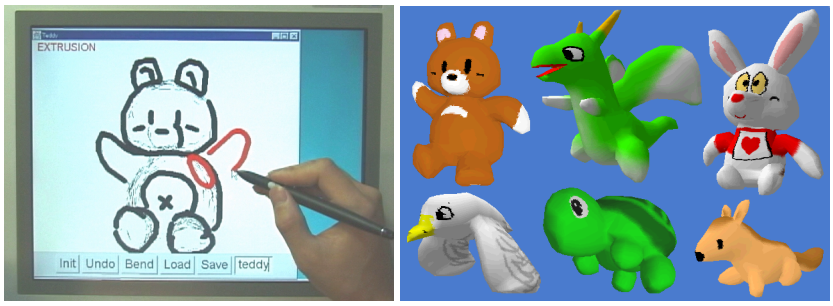

Fig. 6. Teddy in use on a video tablet (left). Example 3D models designed using Teddy (right).

4 Discussions

4.1 Fundamental Limitations of Freeform UI

Freeform UI achieves fluent interaction that is not possible with traditional GUI, but several difficulties are inherent in it. This section discusses three major difficulties (ambiguity, imprecision and learning), and the next section proposes possible solutions to mitigate the problems.

Freeform UI is characterized by its indirect interaction style. Traditional command-based interfaces accept explicit command input and perform the command directly without any hidden processing. In contrast, Freeform UI accepts highly ambiguous freeform strokes as input, and performs complicated processing internally to infer the user's intention from the strokes. The indirect operation is inherently associated with the problem of *ambiguity*. It is difficult to infer appropriate interpretation from the user's ambiguous freeform strokes, and the behavior of perceptual processing can be seen as ambiguous from the user's perspective.

Imprecision is another problem inherent in Freeform UI. While mouse-based careful manipulation of each control point in traditional GUI is suitable for editing precise diagrams, handwritten freeform strokes are not good at precise control. Perceptual processing and informal presentation are also incompatible with precise manipulation.

The indirect nature of Freeform UI also requires a *learning* process by the novice user. Because a simple stroke can transform to a variety of results, the user has to try many strokes and accumulate experience to master the operation. In other words, Freeform UI imposes certain implicit rules to infer complicated information from simple strokes, and the user has to learn the implicit rules through experience.

4.2 Guidelines to Mitigate the Limitations

Based on our implementation and user study experience, we found several techniques and design guidelines to mitigate these problems. Although it is impossible to remove

these difficulties entirely because they are strongly associated with the essential nature of Freeform UI, the following tips work as basic guidelines to design a good Freeform UI system.

First, it is important to give users an appropriate impression that the system is not designed for precise, detailed editing; this will help prevent frustration over ambiguous, imprecise operation. In addition to informal presentation describe in Section 2, a designer can install similar tricks in many places, such as in the introduction to the system, in the system's feedback messages and in the user manuals.

From a technical point of view, construction of multiple alternatives is an effective way to mitigate ambiguity. This strategy is commonly used in Japanese text input systems to type thousands of Chinese characters using a limited alphabet. Pegasus constructs multiple alternatives as a result of beautification and prediction; this feature turned out to be essential to making beautification and prediction perform practically.

As for the problems of learning and ambiguity, it is important to make the interface quick-responding and to ensure that changes can be easily undone so as to encourage trial-and-error experience. For example, Teddy deliberately uses simple algorithms to calculate geometry quickly sacrificing surface quality, instead of using more advanced, time-consuming algorithms. Construction of multiple alternatives is definitely an important feature one should consider when developing a system based on Freeform UI.

Finally, it is necessary to give explanatory feedback for each operation so that the user can easily understand why the system returned the unexpected result. This kind of informative feedback is not very important in traditional command-based interfaces because the system response is always predictable. However, well-designed informative feedback is a crucial feature to prevent frustration and to facilitate the learning process in Freeform UI. For example, Pegasus displays small marks indicating what kinds of geometric constraints are satisfied by the beautified segment. We believe that informative feedback can allow the user to learn how to use the system without having to read manuals or tutorials beforehand.

4.3 User Experience

Although we have a limited amount of user experiences with the prototype systems, it is our future work to obtain further insight by accumulating more experience with real users. Initial user feedback has been quite positive. Users are excited by the demonstrations given by the authors and they successfully start playing around after a minutes of practice. However, the prototype systems are not designed to handle large problems and it is not clear to what extend the Freeform UI approach scales. The scalability problem is actually serious in the Pegasus system; the system generates too many candidates as the diagram becomes complicated. We are currently exploring various ways to solve the problem.

Fortunately, the Teddy system is now widely used as a commercial modeling software and a video game. The users (mostly children) have created various interesting 3D models with them. We believe that the reason for this success is the choice of right application domain: video games do not require precise or large, complicated models which is a perfect match for Freeform UI.

5 Summary

We proposed an interface design framework for graphical computing based on pen-based input, and named it Freeform UI. It uses freeform handwriting strokes as input, recognizes the configuration of the strokes and performs appropriate actions automatically, and presents the result of computation using informal rendering. We introduced four example user interface systems embodying the concept of Freeform UI and discussed its strengths and limitations.

References

1. Black, A.: Visible Planning on Paper and on Screen: The Impact of Working Medium on Decision-making by Novice Graphic Designers. *Behavior & Information Technology*. Vol.9 No.4 (1990) 283-296
2. Goel, V.: Sketches of Thought. The MIT Press (1995)
3. Gross, M.D., Do, E.Y.L.: Ambiguous intentions: A Paper-like Interface for Creative Design. *Proceedings of UIST'96* (1996) 183-192
4. Igarashi, T., Edwards, W.K., LaMarca, A., Mynatt, E.D.: An Architecture for Pen-based Interaction on Electronic Whiteboards. *Proceedings of AVI 2000* (2000) 68-75
5. Igarashi, T., Matsuoka, S., Kawachiya, S., Tanaka, H.: Interactive Beautification: A Technique for Rapid Geometric Design. *Proceedings of UIST'97* (1997) 105-114
6. Igarashi, T., Matsuoka, S., Kawachiya, S., Tanaka, H.: Pegasus: A Drawing System for Rapid Geometric Design. *CHI'98 summary* (1998) 24-25
7. Igarashi, T., Kadobayashi, R., Mase, K., Tanaka, H.: Path Drawing for 3D Walkthrough. *Proceedings of UIST'98* (1998) 173-174
8. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A Sketching Interface for 3D Freeform Design. *SIGGRAPH 99 Conference Proceedings* (1999) 409-416
9. Landay, J.A., Myers, B.A.: Interactive Sketching for the Early Stages of User Interface Design. *Proceedings of CHI'95* (1995) 43-50
10. Markosian, L., Kowalski, M.A., Trychin, S.J., Bourdev, L.D., Goldstein, D., Hughes, J.F.: Real-time Nonphotorealistic Rendering. *SIGGRAPH 97 Conference Proceedings* (1997) 415-420
11. Meyer, J.: EtchaPad - Disposable Sketch Based Interfaces. *CHI'96 Conference Companion* (1996) 195-198
12. Mynatt, E.D., Igarashi, T., Edwards, W.K., LaMarca, A.: Flatland: New Dimensions in Office Whiteboards. *Proceedings of CHI'99* (1999) 346-353
13. Nielsen, J.: Noncommand User Interfaces. *Communications of the ACM* Vol.36 No.4 (1993) 83-99
14. Schumann, J., Strothotte, T., Raddb, A., Laser, S.: Assessing the Effect of Non-photorealistic Rendered Images in CAD. *Proceedings of CHI'96* (1996) 35-41
15. Strothotte, T., Preim, B., Raab, A., Schumann, J., Forsey, D.R.: How to Render Frames and Influence People. *Proceedings of Eurographics'94* (1994) 455-466
16. van Dam, A.: Post-WIMP User Interfaces. *Communications of the ACM* Vol.40 No.2 (1997) 63-67
17. Wong, Y.Y.: Rough and Ready Prototypes: Lessons From Graphic Design. *Proceeding of CHI'92* (1992) 83-84
18. Zeleznik, R.C., Herndon, K.P., Hughes, J.F.: SKETCH: An Interface for Sketching 3D Scenes. *SIGGRAPH 96 Conference Proceedings* (1996) 163-170