Sketching for the Refinement Stage of Design

Gabe Johnson
Carnegie Mellon University
johnsogg@cmu.edu

Abstract

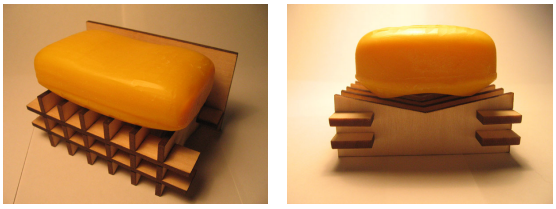
This workshop paper summarizes the motivation for a forthcoming PhD proposal on computational support for sketching during the refinement stages of design. The research explores this space by (1) building a tool empowering people to easily design artifacts by concurrently sketching alongside traditional structured WIMP interaction methods, and (2) evaluating interaction and engineering issues associated with it. Ultimately the contribution of this work will be to further the understanding of interaction techniques for refinement phases of design tools that make use of sketch input.

1 Introduction

Early phases of design can be characterized by idea generation and exploration [2, 16]. Many sketching systems have focused on early phases of design because sketching readily supports such activities. Refinement phases are characterized by incremental revision and production—activities traditional computer design tools support well. However, designers continue to sketch on paper after they have begun revising computer models. These refinement-phase sketches help people solve sub-problems that were not apparent or relevant during earlier exploration.

However, current software design tools are unable to directly leverage these refinement stage sketches. Instead, users manually translate sketches to the computer model. Design tools could be made to understand the user's informal sketching input by leveraging the formal representation already present in the model. It is not clear—from both engineering and HCI perspectives—how a tool that concurrently supports formal and informal input should be made.

The first contribution of this work is the implementation of a design environment called FlatCAD. FlatCAD currently supports users to make models by programming in a domain-specific language called FlatLang [8]. The next version will additionally support structured WIMP operations and sketching input.



(a) (b)

Figure 1. Soap trays programmed in FlatCAD.

The second contribution explores challenges associated with concurrently using informal sketches with formal computer representations. In particular this work focuses on HCI aspects of these tools in order to better understand how sketch-based interaction may be used concurrently with structured WIMP interaction.

The following sections briefly describe FlatCAD, the design environment on which subsequent work will be based. FlatCAD has been chosen as the platform for exploring sketch based interaction because all aspects of the environment are familar to the author and can be changed easily. Technical and interaction challenges associated with adding support for sketching are discussed.

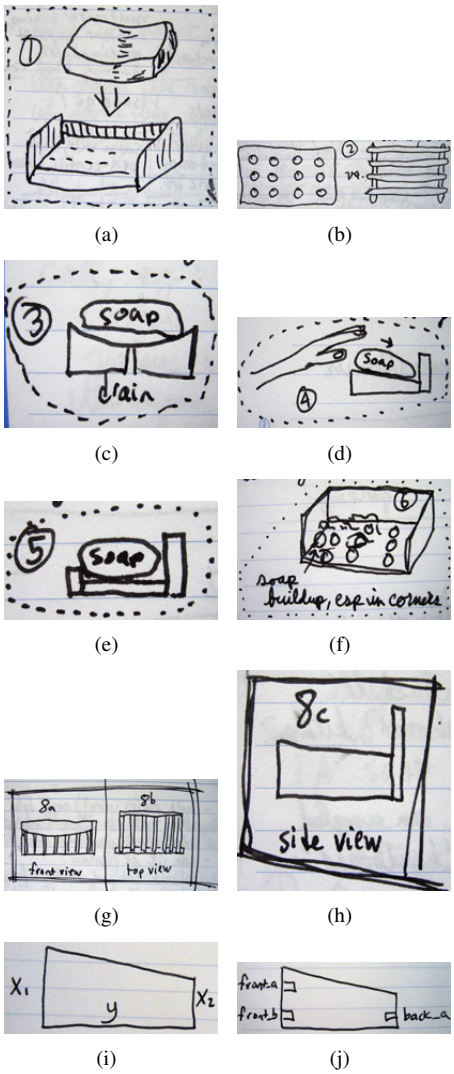
2 FlatCAD Use Scenario

FlatCAD is an environment for designing objects made of flat material for rapid prototyping. Currently, users define shapes by programming in the LOGO-like FlatLang domain-specific language. The output of a FlatLang program is a "cutfile" that is sent to a laser cutter.

FlatCAD has been used to design several classes of physical artifacts, such as construction kits to mechanisms to household goods like soap dishes and toothbrush holders. The following details the process a designer took when making the soap dishes shown in Figure 2.

The designer began by making quick drawings to help think about aspects of making a soap dish. Its primary purpose is to hold a bar of soap, but it also should prevent water

VL/HCC Workshop: Sketch Tools for Diagramming
Herrsching am Ammersee, Germany
15 September 2008
Editors: Beryl Plimmer & Tracy Hammond



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 2. Sketches made during idea generation and exploration.

from pooling, be easy to use, and be stylish.

The first sketches in this sequence supported reasoning about functional requirements, while the last few sketches focused on details of how the soap dish would be constructed and what particular pieces would look like. The final two sketches include parameters and the locations of notches where other pieces adjoin. After these drawings were made, the designer recreated the sketches in FlatCAD by writing a FlatLang program. Figure 3 shows the program's rendering of a single vertical slat.

As work commenced within the CAD tool the designer needed to make several more drawings (Figure 4). Designers often encounter breakdowns that lead them to "go back to the drawing board" in order to restate, review and possi-

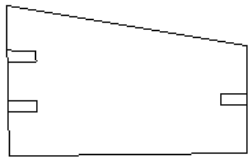
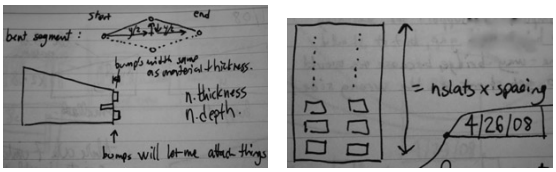


Figure 3. FlatCAD screen shot of a soap dish slat in progress.



(a) (b)

Figure 4. Refinement phase sketches drawn on paper.

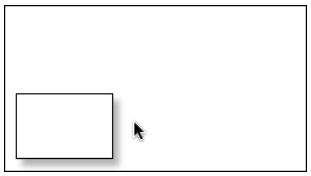
bly reform their general approach for solving a design problem.

Some sketches in the second batch were made to help understand how variables were related, others were used to add needed complexity to the parts so they would fit together well. Even though there was a perfectly good image of the model on the computer screen, the designer had to re-draw it on paper before making progress. Many details already specified in the code (lengths and angles) could not be exactly transferred to the sketch. After making the drawings, the designer had to translate what was learned from the sketch back into the FlatLang code.

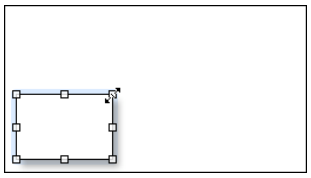
3 Interaction Techniques

In the earlier use case scenario, it would have been helpful if the CAD tool enabled users to sketch directly on the model in order to modify it. The precise nature of the on-screen model provides excellent contextual clues for supporting recognition. For example, several sketches refer to variables that exist in the formal model (e.g. `n.thickness` in Figure 4(a)) or anonymous dimensions (e.g. the vertical edge indicated with the arrowheaded line in Figure 4(b)).

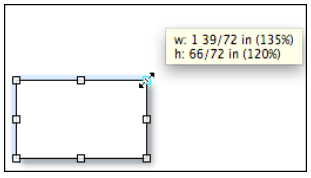
Currently, design tools support input done with a keyboard and mouse in a structured user interface consisting of elements like buttons, pick-lists, dialog boxes, text areas, and so on. Structured interfaces are designed to minimize or eliminate ambiguity of user input. Consider the simple task of enlarging a rectangle. A traditional structured drawing program is likely to provide control handles, which the user



(a) Original rectangle

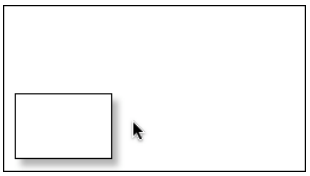


(b) Selecting object reveals 'handles'

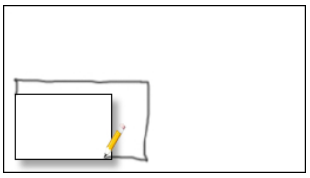


(c) Drag cursor, enlarge to 1/72 inch accuracy

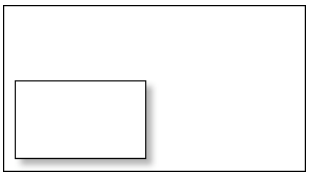
Figure 5. Enlarging an object using traditional WIMP interaction.



(a) Original rectangle



(b) Draw new extents of rectangle, implicitly selecting the particular edges that will change



(c) The exact size is not precisely provided

Figure 6. Enlarging an object using calligraphic interaction.

clicks and drags in order to change an object's size (Figure 5(a)).

An alternate strategy is calligraphic interaction, wherein users draw freehand, ambiguous input [10]. Users may sketch their intention to resize an onscreen box in many ways: drawing arrows indicating what should move, or redrawing the object entirely. Figure 6 illustrates a hypothetical interaction technique where a user resizes a rectangle by redrawing only the portion that must change.

There has been a good deal of research focusing on new types of design tools based on sketching [4, 7, 13]. In addition, many projects have explored isolated interaction techniques for calligraphic interfaces [3, 11, 15]. However, there are currently no widely used conventions for such interaction, and little is known about what constitutes effective day-to-day usage.

One prominent challenge with interactive systems is the "mode problem" [19]: people often are unsure how to enter an editing mode, or are unaware which mode they are in. Structured user interfaces can arguably mitigate this problem more effectively than recognition based interfaces. There are several strategies for handling this in sketch recognition user interfaces, including the Inferred Mode Protocol [17], explicit mode selection triggered by gestures [5, 9], or interactively picking the correct interpretation from a list of several plausible alternatives [6, 15].

4 Engineering

Existing sketching tools produce models for refinement with other design software. In these cases, the connection between the sketching and refining tool has been one way [14]. However, tools that enable iterative sketching and refinement requires two-way connections. For example, SimuSketch recognizes drawings of dynamic systems [12]. The recognized model is passed to the SimuLink program for display and editing. However, modifications made in SimuLink are not reflected in the sketch.

This work seeks to support such a two-way connection between informal and formal modes of expressing intent. Informal sketch input may be provided at one moment, formal commands given the next. The inherent ambiguity of sketch input must be managed because the user may issue formal instructions at any time. Say the user has drawn a shape that may be a rectangle, but may also be an oval. Next the user selects a 'resize' tool and begins to enlarge the shape. It is unclear what should happen at this point.

This is partly an interaction question, but it is also an engineering question. The drawing may provide enough context to disambiguate the shape's identity. This context can come in many forms, such as domain awareness [1], or knowledge of temporal patterns of how people draw [18].

5 Summary

A proposed version of FlatCAD aims to support concurrent informal and formal modes of input during the refinement phases of design. This will serve as a useful means for exploring various human-computer interaction topics related to sketch recognition interfaces. While it is not the focus of this work, the proposed system also provides a foundation to study engineering topics of how to identify, model, and use context for the purpose of recognition.

References

[1] C. Alvarado and R. Davis. Dynamically constructed bayes nets for multi-domain sketch understanding. In *International Joint Conference on Artificial Intelligence*, 2005.

[2] B. Buxton. *Sketching User Experiences*. Morgan Kaufmann Publishers, 2007.

[3] J. Geißler. Gedrics: the next generation of icons. In *Proceedings of the 5th International Conference on Human-Computer Interaction (INTERACT'95)*, 1995.

[4] M. D. Gross and E. Y.-L. Do. Ambiguous intentions: A paper-like interface for creative design. In *UIST '04: ACM Conference on User Interface Software Technology*, pages 183–192, Seattle, WA, 1996.

[5] K. Hinckley, P. Baudisch, G. Ramos, and F. Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 451–460, New York, NY, USA, 2005. ACM.

[6] T. Igarashi and J. F. Hughes. A suggestive interface for 3d drawing. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 173–181, New York, NY, USA, 2001. ACM.

[7] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *ACM SIGGRAPH'99*, pages 409–416, Los Angeles, California, 1999.

[8] G. Johnson. FlatCAD and FlatLang: Kits by code. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing*, 2008 (to appear).

[9] G. Johnson, M. D. Gross, and E. Y.-L. Do. Flow selection: A time-based selection and operation technique for sketching tools. In *2006 Conference on Advanced Visual Interfaces*, pages 83–86, Venice, Italy, 2006.

[10] J. A. Jorge and E. P. Glinert. Guest editor's introduction to "calligraphic interfaces: towards a new generation of interactive systems". *Computers and Graphics*, 24:817–818, 2000.

[11] L. B. Kara, C. M. D'Eramo, and K. Shimada. Pen-based styling design of 3d geometry using concept sketches and template models. In *SPM '06: Proceedings of the 2006 ACM symposium on Solid and physical modeling*, pages 149–160, New York, NY, USA, 2006. ACM.

[12] L. B. Kara and T. F. Stahovich. Hierarchical parsing and recognition of hand-sketched diagrams. In *Proceedings of UIST'04*. ACM Press, 2004. 1029636 13-22.

[13] J. A. Landay and B. A. Myers. Interactive sketching for the early stages of user interface design. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 43–50, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[14] J. Lin, M. Newman, J. Hong, and J. Landay. DENIM: Finding a tighter fit between tools and practice for web site design. In *CHI Letters*, pages 510–517, 2000.

[15] J. Mankoff, G. D. Abowd, and S. E. Hudson. OOPS: A toolkit supporting mediation techniques for resolving ambiguity in recognition-based interfaces. *Computers and Graphics*, 24(6):819–834, 2000.

[16] M. W. Newman and J. A. Landay. Sitemaps, storyboards, and specifications: a sketch of web site design practice. In *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*, pages 263–274, New York, NY, USA, 2000. ACM.

[17] E. Saund and E. Lank. Stylus input and editing without prior selection of mode. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 213–216, New York, NY, USA, 2003. ACM.

[18] T. M. Sezgin and R. Davis. Sketch interpretation using multiscale models of temporal patterns. *IEEE Journal of Computer Graphics and Applications*, 27(1):28–37, 2007.

[19] L. Tesler. The smalltalk environment. *Byte*, 6:90–147, 1981.