Thoughts and Such

Gabe Johnson

July 11 (or so) 2011

1 Fluidity

Fluidity is often cited as one of the properties of pencil-and-paper sketching that makes it such a powerful practice. We say that it is fluid because users may:

1. draw on any suitable location at any time...
2. branch off to new or different ideas at any time...
3. delve into details of particular ideas...
4. simultaneously see earlier drawings...
5. re-draw complete or partial drawings...

Perhaps the benefit of paper sketching is not simply that users enjoy all of the above abilities, but that it *doesn't* have detrimental properties commonly associated with computer tools. Typical computer design software:

1. demands that users provide information that (to the user) is not yet known or relevant
2. has modes that restrict what users may do at any given time or location
3. guides designers to think and work in terms of the tool's worldview

Table 1 is a 2x2 matrix of the positive and negative aspects of paper sketching and computer design software. It would be a victory for designer-centered computer sketching tools to retain the good properties while removing or mitigating the bad ones. The unwanted aspects of each activity are those that break flow. I won't directly address all of them—e.g. I will not be developing a psychic machine with respect to the meaning of sketches "entirely in mind of viewer". Some other aspects (uniqueness of physical sketches) are addressed easily. In the middle are interesting interaction challenges: making it easier to sketch with a computer while also being able to specify via sketching.

My thesis proposal declared that I would build "a coherent set of calligraphic interaction techniques for iteratively making structured drawings that have enough detail that objects

can be fabricated on a laser cutter". This thesis is about *iterative, conversational sketch-based interaction*.

Iterative: Designers arrive at their final products after a series of starts and stops. Along the way, high-level ideas are considered, drawings are made, specifications are given. This does not happen linearly. An idea may be abandoned temporarily as another is recorded and explored, only to be (potentially) resumed later on. Even aborted ideas have value as examples of what is *not* wanted.

Conversational: The proposed system knows something about the domain and the kind of drawings people are likely to make. When appropriate, it will be an active participant in the design process by re-structuring and cleaning up drawings, asking questions about ambiguous parts, and providing contextually-derived help.

Sketch-based interaction: The tool will be based on freehand pen input. This will largely be recognized without needing to be in specific drawing modes (e.g. there will be no "rectangle mode" or "align mode").

2 Elevator Pitch

SkruiFab is a sketch-based system for designing things for fabrication with laser cutters. It helps in the early phases when you're thinking about what to build, and also in the later phases when you're specifying details like lengths, angles, and aligning things. This system is unique in that it lets you bridge the gap between the early and later phases by letting you re-use your sketches, and helps you finish more quickly by intelligently recognizing aspects of your drawing.

3 Roadmap for Software

It would be helpful if I could provide a roadmap that could guild a hypothetical master's student to develop my system. It must be remembered that like all novel systems it is unwise to believe it can be wholly designed first and then simply implemented without change. With that wisdom in mind, the following is a roadmap of SkruiFab.

3.1 Overview

SkruiFab has two primary interaction modes. The first presents an electronic sketchbook where people can draw freely without distraction by the computer. In this mode, the tool is passive. It might silently attempt to recognize certain aspects such as text, or identify unique drawings on a single screen. Whatever recognition is performed in this phase is kept hidden and is only used later at the user's request. The silent mode is used to support free exploration.

The second mode is used to iteratively and interactively specify details of a sketch so that it may be sent to a laser cutter for fabrication. In this phase the system attempts to

recognize what the user is drawing and (upon request) takes action. This phase is conversational: the user 'says' something via sketch input, and the system attempts to understand and responds.

I will focus on the second mode for the moment because it is more complicated than the first.

3.2 Conversational UI for specifying details

SkruiFab's detail speficying-mode is meant involve short sequences of pen input that the system attempts to recognize, and a graphic response. This is similar to a human conversation: one person says something, the other tries to understand what the other has said, and replies. Sometimes the reply is "I don't understand", other times it is to restate the utterance in other words (if the understood meaning is in doubt), or a reply that logically follows.

To summarize, the conversation is composed of three parts that cycle: (1) what the user draws, (2) how the system recognizes the drawing, and to what degree, and (3) how the system responds.

3.2.1 Input Syntax

Many sketch-based systems support particular graphic vocabularies, and often require that people draw in specific ways. SkruiFab will be lenient with the input syntax to allow people to draw as freely as possible. I have examined the kinds of marks people make in laser cutter project sketches. This will form the basis for a domain model used for recognition.

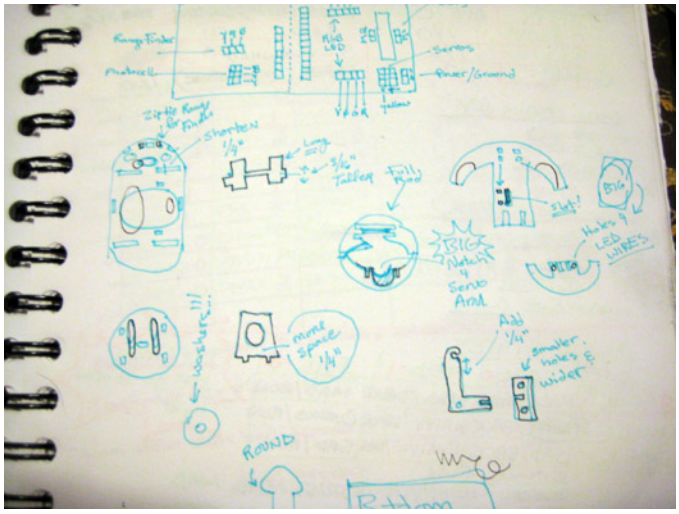

Figure 1: Sketches for a robot built mostly with laser-cut parts.

Figure 1 shows a page from a designer's sketchbook. These drawings were made in the middle of a design process that had already involved physical prototypes. They were made to think about details of robot parts before moving to Adobe Illustrator to create a cutfile. SkruiFab will allow designers to make these kinds of rough drawings without interruption or distraction.

There are a number of distinct drawings on the page. An ink fragment may be part of a stencil boundary, or an arrow (or indicator line), or words, or numeric dimensions. SkruiFab will enable designers to begin with freeform drawings like these and use them as the basis for making a structured cutfile iteratively.

SkruiFab will understand most of the sketch input shown in this sketchbook. It doesn't need to make sense of everything, however. Ink such as the pointy boundary surrounding the words "BIG Notch" does not need to be understood. It *does*, however, need to be possible to tell the system that such ink can be ignored.

The system will understand the following input:

1. Stencil boundaries
    (a) Can represent outside edges or internal edges (holes)
    (b) Boundaries can be lines or curves
2. Arrows that relate one element to another
3. Text and numbers (descriptive names or parameter)
4. Dimension values (lengths and angles)
5. Guides
    (a) Guide lines (dashed or dotted lines)
    (b) Circuilar guides (dashed or dotted lines)
    (c) Reference points (made by drawing thick dots)
6. Hatching to indicate something different about a region
7. Erasure marks to remove/ignore something
8. Overtrace to existing ink to edit it
9. Graphic parameters
    (a) Square bracket to indicate 90-degree angles
    (b) Circular arc inside angle to vaguely indicate angle
    (c) Tick marks on similar length to indicate same length
10. Camera gestures
    (a) Zoom in or out (continuous circular gesture)
    (b) Pan on 2D plane (widget after zoom gesture, perhaps make a pan gesture)

3.3 Recognition

SkruiFab will attempt to recognize input on demand—pressing a keyboard button or activating it via a pen widget onscreen. The system will recognize three broad categories of input:

1. Diagrammatic: 2D graphics, stencil boundaries, modeling operations, arrows, etc.
2. Sentential: Text (parameters, annotations), numbers
3. Unknown: Ink that can not or will not be recognized until explicitly instructed to do so

Diagrammatic recognition will be based on Gestalt principles. Lexemes include lines, curves, dots, and corners; grammar is based on domain semantics and perceptual (and temporal) principles like continuity, colinearity, cotermination, repetition, symmetry, and so forth. Existing information in the model is used to improve recognition. For example, a dashed line is used to indicate a guide. If the drawing already contains reference points, and it is is reasonable to infer the newly drawn guide line passes through them, the exact location of the line is then known.

3.4 Visual Feedback

No recognition system can be 100% accurate. SkruiFab will provide visual feedback to the user in response to recognized input. This cound include coloring or rectifying input, or the system might ask the user to disambiguate or clarify their input in some way.

In the spirit of being a good conversation partner, the feedback should be respectful. If the system were to change things too much, or change things incorrectly, it would distract the user. This would be like somebody interrupting you to finish your sentence and saying the wrong thing.

The feedback should also be simple and consistent, even when wrong. This lets the user working without having think too hard about what the computer is trying to say. The response should imitate the kind of drawn notation people already make. For example, people commonly use a small pair of lines that join at a 90-degree angle near a corner to indicate it is a square junction. This will not only be an expected input phrase, but it will also be used to communicate back to the user. If the system strongly believes a square corner has been drawn, it will reflect this belief by drawing a 90-degree angle symbol in the correct location.

3.5 Tight Loop for Learning

Users will likely find the system somewhat awkward to begin with, partly because they will not be experienced with sketch-based interaction, and partly because the specific visual language is not fully shared between the user and computer.

As noted above the system will give feedback in ways that are graphically similar to what the user might do themselves. This provides a good way to teach the user what the computer expects. Similarly, the system could learn from the user.

	Paper & Pencil	Computer
Good	→ draw wherever → no modes → change to new or different idea at leisure → give only details when known or needed → see other ideas or sketches on page → redraw/erase as desired → be as abstract or detailed as needed	→ interact with typed or recognized objects → can specify parameters and values → modes enable unambiguous actions → gives computed feedback (e.g. critics and compilers) → can run, simulate, print, or manufacture models → good for distributed collaboration → not physically unique
Bad	→ meaning entirely in mind of viewer → physically unique (can be lost/destroyed), must have access to see it → takes skill to make good, accurate sketches → difficult to search large collections of sketches	→ demands information of designer prematurely → modes restrict actions and impose cognitive load → forces users to think in terms of tool's worldview

Table 1: Helpful and hindering properties of paper sketching compared with standard computer design tools.