Does a Sketchy Appearance Influence Drawing Behavior?

Jon Meyer
NYU Media Research Lab
719 Broadway, 12th Floor
New York University
NY 10003
+1 212 998 3381
meyer@cs.nyu.edu

Benjamin B. Bederson
Computer Science Department / UMIACS
Human Computer Interaction Lab
University of Maryland
College Park, MD 20742
+1 301 405 2764
bederson@cs.umd.edu

ABSTRACT

In this paper we examine the role of visual aesthetics in how people interact with computers. Specifically, we are interested in whether simply adopting a sketch-like visual appearance in a drawing application encourages users to interact with the application more freely or rapidly than they would if they were using the standard, precise, rectilinear appearance that most drawing applications now supply.

We carried out two user studies. In the first study, we asked members of the University of Maryland Art History department to draw a series of diagrams using two different line styles. In the second experiment, we used the World Wide Web to collect drawing diagrams from a much broader set of participants. Both studies reveal that subjects draw more quickly using the sketch-like ('wavy') line style than the straight line style.

Keywords

Informal interfaces, sketching, non-photorealistic rendering, rapid prototyping, pen-based computing, aesthetics

INTRODUCTION

If you look at documents you created on your computer several months ago, you may find it hard to separate early drafts from finished results. In digital form, the rough drafts and the finished results often look equally good. Users are forced to rely on rigorous naming conventions and date information to separate versions (see [7] for an exception].

Most authoring tools encourage this visual uniformity. They are designed to produce professional looking finished products, such as might appear in a journal like this. It is difficult to make these tools produce sloppy looking, partially formed drafts that can be immediately differentiated from final drafts or final products.

The features and visual aesthetics adopted by many commercial packages reflect this production-oriented mindset. For example, the WYSIWYG presentation style used by most commercial word processing applications lets the author see up front what the finished appearance of the document will be. The direct-manipulation interaction style [17] in such packages makes it simple to change styles, layouts, fonts and colors. More recent advances, such as the AutoCorrect features of Microsoft Word, let users see and correct typos as they make them.

At first glance, this is a good thing: documents always look very pleasing, and have fewer typos. However, for some tasks, such as creating a sketch or quick draft, a precise appearance may not be appropriate. With production oriented tools, it can be hard to overcome the urge to tidy the appearance of an electronic document, fixing small errors as they occur.

We conducted an informal survey, asking members of an office whether they had ever felt that they had spent too long tidying up a finished document on a computer to make it appear "just right" on paper. Four out of five members of the office said they had. For at least some of these documents, a more approximate solution would probably have sufficed, but the software in use was not designed with approximations in mind.

There is increasing evidence that rough, sketch-like representations, with all their flaws, are an important part of early design. Sketches offer an important mediating representation [22]. They invite playful thinking in the creator [4,1], and convey important information to the reader [16]. In a survey of graphic designers, Black [1] summarizes the difference between screen generated and hand-rendered design drafts as follows:

> "Hand-rendered drafts vary in appearance from very provisional to very final... The different appearances of hand-rendered drafts give immediate cues to the stage of development of a solution and the different qualities of draft focus attention on particular aspects of a solution… The finished appearance of screen-produced drafts shifts attention from fundamental structural issues. Since the text looks so finished so

soon, it can appear that little further work is needed and users, especially novice users, may be distracted from experimenting fully with visible form." [1, p.286].

Similarly, Wong [21] notes that designers use low-resolution thick markers during prototyping to de-emphasize aspects of a design that are not important.

Black and Wong imply that prototyping software may benefit from low-resolution imprecise-looking tools, which do not offer such a fine level of control. One interesting question is the extent to which appearance alone (as opposed to, say, input device or other factors) is important in these kinds of tools.

In this paper, we first outline related work. Then we describe two experiments - a formal study and a web-based study - looking at the influence of line style on a series of drawing tasks. We present results which show that a precise, straight-line appearance does cause users to spend more time working on a drawing than a wavy, sketch-like appearance. Finally, we conclude with some suggestions for developers of authoring and prototyping environments.

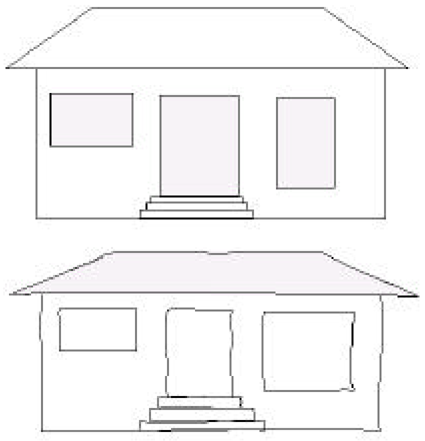
Figure 1: A house drawn using two different line styles. Note that both drawings are constructed using the same number of shapes and vertices. However, in the second diagram, JavaDraw adds noise to each line segment to create a sketch-like appearance.

RELATED WORK

Recently, there has been a modest increase in interest in the topic of aesthetics in Human Computer Interaction (HCI). For example, Tractinsky [20] studies the effect of aesthetics in different cultural settings, validating the results of a study by Kurosu and Kashimura [10]. Tractinsky observes that mainstream HCI often belittles or ignores aesthetics, even though both studies show that aesthetics plays an important role in usability. Our work extends these studies by examining aesthetics in computer drawing programs.

Computers have been used for drawing since the original Sketchpad [18], and numerous drawing and display devices have been proposed. However, there has been little work studying the effect of line style on performance in these kinds of packages.

Readership issues of sketch-like representations were studied in [16]. The authors observed that architects often trace over computer output to create drafts with a hand-sketched look, because they feel that such sketches invite more appropriate feedback from clients during the early stages of a project. Similarly, Wong [21] describes how detailed designs invite less creative feedback from team members commenting on the design, because they focus on the details rather than issues. Our work differs from these studies because we examine user behavior during sketching, rather than readership behavior that takes place after the sketch is complete.

Citrin [2] looks at user performance using different types of diagram editors. In his study, he compared gesture based interaction techniques with palette-based interaction techniques, using a pen as an input device. In our study, we look only at palette based systems, using a mouse as an input device, and consider the effect of line style in these systems.

There have been a number of studies of the role of sketching in prototyping. Black [1] compared the working methods used by graphic designers when they plan documents using computers and paper, and concluded that students working on computers compromise their design ideas more than those working on paper. Landay [11] conducted an informal survey of interface designers, and found that many of them used rough paper prototypes.

Several researchers have developed pen-based systems to support sketching and ideation (e.g., [11, 6, 9, 15]). These systems all aim to support rapid prototyping by providing an intuitive input device (the pen) and very flexible user interfaces. We are interested in applying the sketch-like aesthetic found in these systems to a broader range of applications, including mouse-based applications and applications that have traditionally had a very precise appearance (e.g., CAD or visual programming).

A number of people have explored alternative techniques that allow users to create drawings using a computer - including beautification systems (e.g., [8]), demonstrational interfaces (see [3] for a survey) and constraint-based techniques (e.g., [23]). A major goal of these systems is to convert high-level input from the user and convert it into sketches. Such approaches represent valuable interaction techniques for rapid prototyping and design, though several of the systems opt for a traditional, precise-looking appearance in their interface, rather than a more sketch-like appearance. A notable exception is Sketch [23], which supports 3D editing using constraints, and offers a non-photorealistic appearance.

Several groups have built systems that offer non-photorealistic rendering styles (e.g., [14]). We are not aware of any user performance studies comparing standard and non-photorealistic rendering styles.

DRAWING SOFTWARE

In this paper we report on two user studies - both using the same Vector-based drawing applet, called JavaDraw, which was written specifically for testing purposes. In this section we describe the features of the JavaDraw application.

JavaDraw supports simple drawing tools for drawing lines, rectangles, ovals, and polygons. There is a selection tool for selecting, moving, resizing and reshaping objects. There are a set of command buttons to perform cut/copy/paste/delete operations, and also front/back operations. JavaDraw was designed to be very familiar to people who use drawing packages regularly, and has an interaction model similar to Adobe Illustrator. It is operated using just a mouse (no keyboard support was included). JavaDraw includes an event logging mechanism, that records all user interactions. We developed tools to support playback and analysis of this drawing data.

We omitted many features found in commercial drawing packages. We did not include text (since it was not required for any of the tasks we used). We also only included one fill color – users could either draw unfilled shapes, or shapes filled with gray.

Notably, JavaDraw does not support straight-line constraints or grid snap. We discounted constraints for several reasons:

- While constraints do help users construct certain classes of diagrams quickly, they do not generalize to support all drawing tasks.
- Some people do not use constraints. We wanted to remove constraints as a confounding variable in our study.
- For sketching tasks, our goal is to discourage users from spending time concentrating on presentation and precision. For these tasks, constraints, grids and other construction aids may not be appropriate.

Line Style

JavaDraw supports two different line styles: straight and wavy. We experimented with a number of different wavy line styles. Our eventual choice of style for this study was dictated by a number of factors:

- We wanted a line style that could be implemented efficiently in Java (this ruled out anti-aliasing, thick lines, finely curving lines, and lines which varied in width over their length).
- We felt the line style should be very similar to the standard 1 pixel straight line, so that a comparison with straight lines could be made – this ruled out feathered lines, lines which are “over-stroked”, and lines which extend beyond the specified start/end points.

The line style we adopted is shown in Figure 1. It is the same approach used in [12]. The line algorithm divides each line vector into 10 segments, and adds Perlin noise [13] to the vertices of each segment to produce a wavy looking line. The amplitude of the noise varies in proportion with the length of the line, but is capped at 50 pixels to avoid overly wavy lines. Various parameters can be tuned, including the base amplitude (noiseAmp) and frequency (noiseFreq) of the noise. We used an amplitude of (0.05 * lineLength) and a frequency of 0.1 for our formal study. Our line algorithm is shown below:

```
void drawWavyLine (int ax, int ay,
                   int bx, int by) {
   int lx = bx – ax, ly = by - ay;
   int lastX = 0, lastY = 0, x1 = ax, y1 = ay;
   double len = sqrt(lx*lx+ly*ly);
   int amp = (int)(min(len * noiseAmp, 50));

   double wx = (((-ly * amp) / len))/2,
          wy = (((lx * amp) / len))/2;

   for (int i = 1; i < 10; i++) {
       int x2 = (int)(ax + (t/10.0)*(bx-ax) +
                noise(p) * wx);
       int y2 = (int)(ay + (t/10.0)*(by-ay) +
                noise(p) * wy);
       g.drawLine(x1, y1, x2, y2);
       p += noiseFreq;
       x1 = x2; y1 = y2;
   }
   g.drawLine(x1, y1, bx, by);
}
```

A SIMPLE MODEL OF DRAWING

When people draw using a computer, they spend some of their time thinking or planning their next move, and some of their time actually executing a stroke or manipulating the graphics. In our simple model of drawing, we assume that people do not interleave these activities. During planning, users decided what their next stroke should be used for. Then, during the execution phase, they are estimating distances, and comparing the line they have drawn with their mental model of how the drawing should progress. This is a simplified model since it does not account for people who carry out these activities in parallel. Also, it does not take into account the experience or problem-solving skills of the drawer. Finally, we ignore the emergent and dynamic aspects of sketching [15, p.175][19] and other cognitive factors in this model.

Using this model, we predict that, when drawing with wavy lines, users will spend less effort during the execution phase of their drawing, compared with straight lines. That is, with wavy lines they will spend less effort focusing on precision as they draw, and therefore they will draw individual strokes more quickly – leading to faster execution times. To test this, we started with one main hypothesis:

Hypothesis 1. The task completion time will be less for wavy lines than it is for straight lines

We base hypothesis 1 on the assumption that users draw strokes more quickly. We can verify this using a second hypothesis:

Hypothesis 2: Users will draw individual stokes with greater velocity with wavy lines than with straight lines.

Finally, we are interested in whether line style influences the planning and thinking phase of drawing. For the simple tasks used in this study, we expect that the line style will not effect planning and thinking. To test this, we add a final hypothesis:

Hypothesis 3: Users will spend the same amount of 'idle time' (not interacting with the program) with wavy lines compared to straight lines.

STUDY 1: FORMAL STUDY

The Subjects

In the first study, the subjects were 18 volunteers from the Electronic Media Center in the Art History department at the University of Maryland. Subjects were paid $5 for their participation in the study, which took about 40 minutes.

We gave subjects a written questionnaire asking for background information, then filtered subjects, selecting those who had spent more than a few months working regularly with programs such as Illustrator and Photoshop (all 18 subjects met these requirements).

Subjects included undergraduate students and lecturers; 9 were Fine Art or Art Studio majors, 2 were Art History majors, 3 were lecturers in Fine Art, and the remaining 4 subjects were taking art courses, but not majoring in art.

The mean reported age of subjects was 26 (the oldest was 45 and the youngest, 20). 12 (66%) were male and 6 were female. Only one subject was left-handed. When asked whether they considered themselves primarily artistic or technical, 15 (83%) said artistic, and the remaining 3 (17%) said technical.

The mean experience using computer drawing programs was 2.5 years, though one subject had been using computer drawing programs for 12 years, and three had only used drawing programs for a semester. When asked how often they used drawing programs, given the choices "never", "rarely", "sometimes", "often", "every day", all of the subjects responded either "often" or "every day". When asked if they agreed that they felt comfortable drawing using a mouse, 4 subjects disagreed, 4 were neutral, and 10 agreed (5 of them strongly agreed). The mean response lay between neutral and agree. The subjects were also asked to rank their computer drawing skills, on the scale "Novice", "Fair", "Satisfactory", "Good", "Expert". Half of the subjects considered themselves "Good" to "Expert", and the other half "Fair" to "Satisfactory".

Method

We used a fully crossed factorial design with repeated measures. Each subject was asked to draw five drawings using one line style, and then the same five drawings using the other line style. Half of the subjects used straight lines first, then wavy. The other half drew with wavy lines first, then with straight lines. The order of the five tasks within each section was also randomized. We decided to separate the experiment into two sets of five tasks so we could measure interactions between the two line styles. All subjects used the same IBM Thinkpad 600 computer, with a 12.1" display, using a two button mouse set to a slow speed with no acceleration.

Before starting the experiment, each subject was given a short briefing. They were asked to read a two-page document describing the JavaDraw application, and were allowed to ask questions. They were also given written instructions explaining that they would be creating two sets of five drawings, using two different line styles. They were instructed to try to emphasize both speed and accuracy in their drawings – not to spend too much time on each task, but at the same time fulfilling the given instructions. Finally, they were told that "the drawings should be suitable to show to a colleague as a draft of a design idea. They need not be perfect, though they should follow the instructions, and be reasonably accurate. "

Tasks

There were five different drawing tasks:

1. Draw a 5x5 grid of a specified size
2. Draw a house, including some windows and a door
3. Draw a circle, then shade in the circle using 5 or more horizontal hatch lines
4. Draw an "org chart" (a binary tree with 7 nodes).
5. Draw a cube, showing just the top, front and left faces.

These tasks were designed to be easy to comprehend and also to test a range of drawing activities. The grid task and the circle task were designed to test how much effort subjects spent trying to align horizontal or vertical lines to remove aliasing artifacts such as kinks and 'jaggies'. The "org chart" task aimed to explore how much effort subjects spent moving and adjusting a flow diagram layout - such as you might find in a visual programming language. The house task explored how subjects performed given an open-ended activity. The cube task explored the effect of line style on perspective drawing.

For each task, subjects were first given written onscreen instructions, which included hints on how to complete the task. For example, for the grid task, the instructions were:

> **Draw a 5x5 grid (i.e., 5 squares across and 5 down) Hint: Use the rectangle tool to draw the border of the grid, and the line tool to draw grid lines. You can use Copy and paste to copy the grid lines. Make the grid fit in the size of the guide shown in magenta in the JavaDraw window.**

After reading the task descriptions, users clicked on a "Start Drawing" button and were presented with a new JavaDraw window to draw in. Task timings were measured from the time of the first interaction with JavaDraw, extending until the user clicked on a "Done" button.

We conducted an informal pilot study (n=4) to test the task instructions and the usability of the JavaDraw program, and made several refinements to the tasks and the software based on the pilot study. Because of the basic nature of the tasks and the JavaDraw drawing software, very few subjects had difficulty comprehending or completing the drawing tasks (although many subjects incorrectly included the right or back faces on their cubes). When there were comprehension problems, subjects were allowed to ask questions about the task before they started drawing. Subjects were also allowed to ask questions about the software, which we felt was acceptable since we were not testing the software for learnability.

Results

Hypothesis 1: Users spend less total time on each task

Subjects completed drawing tasks faster on average when using wavy lines than when using straight lines ($F_{4,18}$ = 15.9, p=0.001). This was particularly notable on the grid task, where the mean completion time for wavy lines was 149 seconds, compared with 213 seconds for the straight lines.

The time to complete the house drawing task is apparently not faster when using the wavy lines than straight lines. The task completion times for this task are probably not a useful measure, since the task was open-ended and each subject executed the task very differently.

Further analysis reveals that the ordering of line styles is a significant factor ($F_{1,18}$ = 58.0, p=.001). That is, subjects whose first five tasks were done using wavy lines worked more quickly on their second set of five tasks than those who started with straight lines. This implies that users can learn to work with less precision using straight-line tools.

Task time (secs)		Grid	House	Circle	OrgChart	Cube
Wavy	Mean	148	179	107	147	83
	σ	*66*	*119*	*51*	*79*	*46*
Straight	Mean	213	175	123	189	108
	σ	*100*	*90*	*56*	*88*	*69*

Table 1: Mean time to complete tasks (n=18)

To gain a ballpark comparison figure for the level of difficulty of the drawing tasks used in this test, we asked five people chosen at random in our lab to complete the five drawing tasks using pencil and paper. The mean completion times for the tasks are shown below.

Pencil & Paper	Grid	House	Circle	OrgChart	Cube
Mean time (secs)	15	26	16	25	10

Table 2: Mean times to complete tasks (in seconds) using pencil and paper (n=5).

It is clear that, even using wavy lines, vector based drawing programs like JavaDraw are approximately 8 times slower than using pencil and paper.

Hypothesis 2: Users will draw strokes more quickly

Looking at the mouse speed during drawing (i.e. how many pixels the mouse was moved per second during drawing), we found that subjects drew significantly quicker (p=.01) with wavy lines. For example, on the grid task, subjects drew 160% faster with wavy lines than with straight lines. Note that mouse speed is higher for all tasks, including the House task. This figure is perhaps more reliable than the task completion time, since it measures how users draw individual strokes and therefore factors out any differences in approach and planning that the subjects used.

We observed that one of the more expert users drew with approximately the same speed with both line styles, although he did express a preference for the wavy line style. Perhaps trained experts are able to overcome the temptation to tidy sketches, and therefore do not draw more quickly with sketchy line styles.

Speed (pixels/sec)		Grid	House	Circle	OrgChart	Cube
Wavy	Mean	55	68	91	60	71
	σ	*31*	*39*	*50*	*24*	*36*
Straight	Mean	33	57	77	55	52
	σ	*14*	*25*	*29*	*18*	*30*

Table 3: Mean speed of mouse during drawing, in pixels per second (n=18).

Hypothesis 3: Users will spend same amount of 'idle time'

We analyzed the proportion of time that users spent interacting with the program, compared with the proportion spent 'idle' (not interacting with the program).

We did not find a significant difference in idle time between for the two different line styles. However, we also did not find a correlation between the two line styles. For example, for the grid task, the Pearson's correlation was .02, (p=.94). It may be that our measure of idle time is inaccurate – or that a more complex account is needed to explain for how users spend their time during drawing.

Idle time (secs)		Grid	House	Circle	OrgChart	Cube
Wavy	Mean	65	63	64	67	59
	σ	*7*	*6*	*8*	*8*	*16*
Straight	Mean	62	60	64	67	59
	σ	*5*	*25*	*7*	*6*	*12*

Table 4: Percentage of 'idle' time – i.e. time not spent interacting with JavaDraw (n=18).

Evaluation Questionnaire

After performing the drawing tasks the subjects were asked to complete a written questionnaire. The first three questions asked about the JavaDraw drawing application, with no consideration of the line style used; the subjects were asked to respond to the statements "I enjoyed using the drawing application", "The drawing application was easy to use", and "I was able to draw what I envisioned", using a scale from "Strongly Disagree" to "Strongly Agree". Next, subjects where asked questions regarding the line style. Using the same scale, they responded to the statements: "I liked the wavy line style" and "I found myself drawing differently using the two line styles". Finally, subjects were asked which line style they would prefer to use for concept diagrams and early prototypes, and to describe any difference they experienced in words.

Results from the Questionnaire

Table 1 summarizes the responses to the first five questions. Most subjects enjoyed using the drawing program, and found it easy to use. They also felt that they could draw what they envisioned.

Regarding the two line styles, most subjects agreed that the change in line style caused them to draw differently. Only three subjects felt that there was no change in drawing behavior with the two styles. Of those three subjects, two later agreed that they did draw more quickly with wavy lines, but they felt that they used the same approach to drawing with both straight styles.

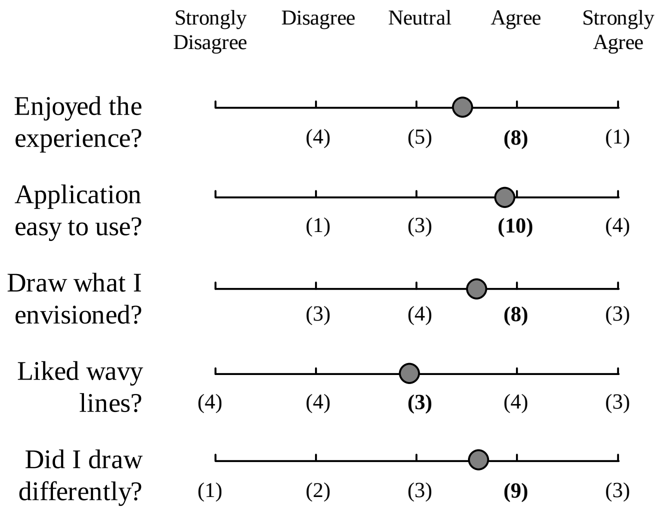

Table 5: Summary of responses to questionnaire. The gray circles indicate the mean response, on the scale "Strongly Disagree" to "Strongly Agree". The numbers in parenthesis indicate how many subjects gave each response. The most frequent response is indicated in bold.

When asked to describe the difference between the two styles in words, subjects were divided. Six subjects gave positive feedback. One person in this group liked the wavy lines, saying "When designing, perfection is not always the key. Errors can be a good thing and you can make them work for you. Using the wavy lines can add an element of surprise, making the design more human." Two other subjects in this group said they felt "more relaxed" using the wavy lines.

On the negative side, one subject said "the wavy lines were different from what I am used to in other applications, so it took me some time to get used to. When I use a computer to sketch I expect nice straight lines. The sketchy lines were annoying to predict." Other subjects said that they cared less about their drawing; that the line style was distracting, and that they wanted more control over the appearance of the diagram.

When the subjects were asked which line style they would prefer for concept diagrams and early sketches, only five (28%) voted for the wavy lines. Coupled with the low rating that subjects gave the wavy line style, this indicates that more work is needed to develop a sketchy line style that appeals to artists.

One source of frustration in this test may have been that subjects had no control over which line style to use, or over the appearance of the sketchy lines. If they had selected the line style from a palette they may have felt more positive about the style.

In subsequent discussions with the subjects regarding the goals of the study, two subjects re-evaluated their opinions of the line style once they understood the purpose of the test. Saying that they thought that a sketch mode was a good idea. This indicates that education is important to teach users the value of sketch-like representations.

STUDY 2: WEB-BASED STUDY

The World Wide Web offers interesting opportunities for user testing – notably, it represents a very large population of potential subjects, and therefore an easy means of collecting large quantities of data. Of course, carrying out experiments using this population has serious drawbacks: there is less control over selecting subjects and over experiment conditions. Also, the subjects may be dishonest or may complete the experiment incorrectly, or they may lose interest and stop the test before finishing.

Some researchers have already used the Web to collect user preference data (typically using questionnaires), or have examined usability of Web documents. We believe that there are many forms of user testing which can benefit from web-based studies – especially studies that replicate a more controlled formal study, and early explorations to gain feedback about user interface concepts.

Utilizing technologies such as Java and the Internet, it is possible to write small applications that run in people's web browsers, and record data on how web users interact with those programs. For our Web based study, we modified the JavaDraw program so that it would run in a web browser. We then ran an experiment using the same five tasks as the formal study, except that we adopted a between-subjects design, asking each subject to complete five diagrams using one line style. The ordering of the drawing tasks was randomized. After completing each drawing task, subjects were asked if they wished to perform another task. Most subjects completed one or two drawings and then quit out of the test.

We used three different line styles in the web based test: the straight line style, the same wavy line style we used in the formal study, and a third wavy line style, which used the wavy line algorithm described earlier, but with a larger noise amplitude. We utilized a web technology called "Magic Cookies" to record which line style each subject used, and to give each participant a unique ID. This way, each subject saw only one line style, even if they repeated the test.

Using email, we invited colleagues and friends from the design community to participate (sending them a URL and a brief description of the nature of the experiment). We also sent email announcements to several news groups (including the AutoCad news group and several Art groups).

Results

We collected 330 drawings from 221 different subjects. 62 subjects completed the online questionnaire. We then filtered the results, discarding incomplete or incorrect drawings, or drawings that took less than ten seconds or more than a thousand to complete. Many of the drawings submitted by people responding to the newsgroup postings did not meet the acceptance criteria, and around 70% were rejected. The acceptance rate from the targeted emailing was much higher (approximately 80% were accepted).

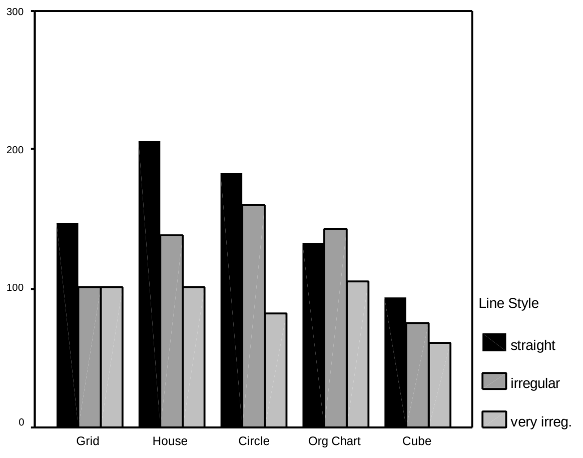

Figure 2: Estimated mean time in seconds to complete tasks, web-based study.

Running a MANOVA analysis of this data, we discovered that the results are consistent with the data recovered for the formal study, indicating that the effect may generalize to a wider population. For example, looking at the mean time to complete tasks, we again discovered that line style was a significant factor (p=.001), where subjects with wavy lines completed tasks more quickly than those with straight lines. Furthermore, users with very wavy lines completed tasks more quickly than those with wavy or straight lines (see Figure 2).

Because of space constraints (as well as the approximate nature of the data), we have not included the full statistical analysis of the Web Study here. Readers can visit http://www.cat.nyu.edu/drawing_study to see the data collected from this study, and also to run the JavaDraw drawing and analysis programs.

DISCUSSION

The results presented in this paper clearly indicate that visual appearance does play a significant role in how people interact with software. Furthermore, they show that people draw more slowly when given a precise line style than they do with a more sketchy line style. These results are not too surprising – since if you don't let users draw straight lines, it seems logical that they will spend less effort trying to.

We expected the wavy line style to be more popular than it was. Less than a third of the subjects said they would choose the wavy line style for their sketches. More research is needed to discover what approximate line styles appeal to artists. For example, antialiased lines may offer the same speed advantages and also have more visual appeal.

Based on these results, we can make several recommendations to application developers:

1. It is important to try different visual styles in the application – especially in authoring and prototyping environments. Some visual styles may cause users to spend too long focussing on appearance and detail. Sketch-like displays may help users to focus more on the task at hand.
2. Education is important to teach computer users about the value of rough, sketch-like representations.
3. Care (and user testing) is needed when selecting a sketch-like visual appearance. Users should be given the choice to disable or modify the appearance.

CONCLUSION

The results presented in this paper are only a small step towards learning more about one of the cognitive factors involved in drawing and sketching. Research is needed to discover the longer-term effects of sketch-like representations, and also to build a better model of how people sketch.

Based on the results of this study, we believe that computer-based tools should support a range of visual affordances - from highly refined to quick rough sketches. Furthermore, computer tools should be able to take material created in one stage in the creative process and reuse it in the next. However, care needs to be taken when designing the appearance of such tools.

ACKNOWLEDGMENTS
This paper would not have been possible without the generous support of Ken Perlin. We would also like to thank Nare Ratnapala, Catherine Hays, Duane Whitehurst, and Therese Stowell.

REFERENCES

1. Black, A., “Visible Planning on paper and on screen: The impact of working medium on decision-making by novice graphic designers”, *Behaviour & Information Technology* Vol. 9, No 4, pp. 283-296, 1990.

2. Citrin, W., “Requirements for Graphical Front Ends for Visual Languages”, *IEEE Symposium on Visual Languages*, pp. 142-150, Bergen, Norway, August 1993.

3. Cypher, A., “Watch What I do: Programming by Demonstration”, MIT Press, Cambridge, 1993.

4. Edmonds, A., Moran, T., “Interactive Systems for Supporting the Emergence of Concepts and Ideas”, http://bashful.lboro.ac.uk/chi-wshop/, *Conference on Human Factors in Computing Systems* (*CHI’97) One Day Workshop*, ACM, New York, 1997.

5. Fekete, J.D., Bizouarn, E., Cournarie, E., Galas, T., Frederic, T., “TicTacToon: A Paperless System for Professional 2D Animation”, *Proceedings of SIGGRAPH ’95*, ACM, New York, 1995.

6. Gross, M., Do, E. Y., “Ambiguous Intentions: A Paperlike Interface for Creative Design”, *Symposium on User Interface Software & Technology (UIST ’96),* pp. 183-192, ACM, New York, 1996.

7. Hill, W., Hollan, J., Wroblewski, D., McCandless, T., “Edit Wear and Read Wear Text and Hypertext”, *Conference on Human Factors in Computing Systems* (*CHI’92),* p.3-9, ACM, New York, 1992.

8. Igarashi, T., Matsuoka, S., Kawachiya, S., Tanaka, H., “Interactive Beautification: A Technique for Rapid Geometric Design”, *Symposium on User Interface Software & Technology (UIST ’97),* pp. 105-114, ACM, New York, 1997.

9. Kramer, A., “Translucent Patches - Dissolving Windows”, *Symposium on User Interface Software & Technology (UIST ’94),* pp. 121-130, ACM, New York, 1994.

10. Kurosu, M., Kashimura, K., “Apparent Usability vs. Inherent Usability”, *Conference on Human Factors in Computing Systems (CHI ’95) Conference Companion*, pp. 292-293, ACM, New York, 1995.

11. Landay, J., Interactive Sketching for the Early Stages of User Interface Design, Ph.D. Thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA., 1996.

12. Meyer, J., “EtchaPad: Disposable Sketch Based Interfaces”, *Conference on Human Factors in Computing Systems* (CHI ’96), *Conference Companion*, pp.195-196, ACM, New York, 1996.

13. Perlin, K., “An Image Synthesizer”, *Computer Graphics* 19, 3 (July 1985).

14. Salisbury, M., Amderson, S., Barzel, R., Salesin, D., “Interactive Pen-and-Ink Illustration”, *Proceedings of SIGGRAPH ’94*, ACM, New York., 1994.

15. Saund, E., Moran, T., “A perceptually-supported sketch editor”, *Symposium on User Interface Software & Technology (UIST ’94),* pp. 175-184, ACM, New York, 1994.

16. Schumann, J., Strothotte, T., Raab, A., Laser, S., “Assessing the Effect of Non-Photorealistic Rendered Images in CAD”, *Conference on Human Factors in Computing Systems* (CHI ’96), pp.35-41, ACM, New York, 1996.

17. Shneiderman, B., “Direct Manipulation: A step beyond programming languages”, Computer, 57-69, 1983.

18. Sutherland, E., “Sketchpad: a Man-Machine Graphical Communication System”, Proceedings of AFIPS Spring Joint Computer Conference (23), pp. 329-346, 1963.

19. Tann, M., “Saying what it is by what it is like - describing shapes using line relationships”, *The Electronic Design Studio*, McCullough, Mitchell and Purcell (eds.), pp. 201-214, MIT Press, Cambridge, MA, 1990.

20. Tractinsky, Noam, “Aesthetics and Apparent Usability: Empirically Assessing Cultural and Methodological Issues”, *Conference on Human Factors in Computing Systems (CHI ’97),* pp. 115-122, ACM, New York, 1997.

21. Wong, Y. Y., “Rough and Ready Prototypes: Lessons from Graphic Design”, *Conference on Human Factors in Computing Systems (CHI ’92), Posters and Short Talks*, pp. 83-84, ACM, New York, May 1992.

22. Wood, C., “The Cognitive Dimensions of Mediating Representations”, *Conference on Human Factors in Computing Systems* (*INTERCHI ’93) Adjunct Proceedings*, pp. 155-156, ACM, New York, 1993.

23. Zeleznik, R., Herndon, K., Hughes, J., “SKETCH: An Interface for Sketching 3D Scenes”, *Proceedings of SIGGRAPH ’96*, pp. 163-170, ACM, New York, 1996.