Metamouse on trial: confessions of a wanton turtle

David L. Maulsby and Ian H. Witten

Department of Computer Science
University of Calgary
Canada T2N 1N4

email: maulsby@cpsc.ucalgary.ca, ian@cpsc.ucalgary.ca

Abstract

We conducted a usability study of Metamouse, a demonstrational interface to a graphics editor that infers complex constraints in a procedural paradigm using graphical construction. Aspects of its inference mechanism and metaphor were tested by having a variety of users perform standard tasks with and without its assistance. We found that people learn to use static rather than dynamic constructions, and the system fails to learn some task decompositions. In particular, its rules for inferring iteration over a set of objects are both inadequate and inadequately disclosed by the metaphor. To address the problem, we propose an explicit representation of sets and generalization over multiple examples.

Keywords: demonstrational interfaces, user testing

1. Introduction

Metamouse is a demonstrational interface to a simple drawing program [11,13]. The user specifies a procedure by performing an example execution trace, manipulating graphics primitives as both data and tools to express constraints. The system generalizes the user's action sequence, identifying key features of individual steps and disregarding coincidental events. It creates a program with variables, loops and branches, which it uses to predict upcoming actions, thereby reducing the tedium of repetitive and precise graphical editing. It uses default reasoning about graphical constraints to make initial generalizations, and enables the user to correct these either by rejecting predictions or by editing iconic tacks which it displays after each action.

This paper reports a user study which was designed to reveal shortcomings in the Metamouse system and provide information about the expectations of typical users that could be used to improve the design of future programming-by-example systems. One of the challenges of user interfaces that employ inference is to equip users with a suitable metaphor on which they can base their model of the system, and we were particularly keen to expose mismatches between user's expectations of what Metamouse should be able to learn and what it could actually do. We were not disappointed!—the study revealed a host of shortcomings and provided a wealth of information that will be used in the design of future versions.

2. Metamouse: PBE in a graphical domain

2.1. Metamouse's computational model

A demonstrational interface can be characterized by the model of end-user computation it represents. Most research has focussed on procedural constructs [1,6,7,8,9,10,16,17, 21], although some researchers have adopted declarative approaches [4,17,18,19]. Metamouse was intended to bridge the gap by representing simple constraints declaratively (touches) and more complex ones procedurally (graphical constructions). Sets or categories of objects defined by usage are also constructed procedurally, by evaluating selector functions at each step of an iteration. These functions distinguish pattern matching from object re-use, enabling Metamouse to learn multi-stage edits that both break and restore constraints among objects. In contrast, Noddy [1] and EAGER [6] select only by pattern matching. Peridot [17] and ETAR [8] iterate over user-declared sets. In the user trials we wanted to find out whether people could proceduralize constraints, and whether our selector functions were adequate to model object usage. A subtle point here is that selector functions influence the way a procedure can be structured, by limiting the formation of loops.

2.2. Metamouse's metaphor

No interface that uses inference can be reliable unless the user understands its bias—that is, the range of alternatives it considers—well enough to predict its responses [15]. When the bias is simple or nicely fits user expectations as guided by a metaphor, experience of the system's responses reveal it soon enough. The effectiveness of metaphor was a crucial issue in our user study.

Metamouse has a rather elaborate metaphor because its bias excludes so much of what users might expect from a drawing assistant. These limitations are explained in a one-page

informal description. Aspects of the metaphor were tested previously with a questionnaire based on snapshot views of Basil touching and manipulating objects [12].

Constraints

The metaphor explains that Basil is touch-sensitive but has limited vision, since he crawls about the screen and therefore lacks a global view of its contents. Constructions help him sense relations by feel. This bias is revealed by showing a tack at every touch observed, black indicating constraint, white otherwise. The user can toggle these. The questionnaire [12] established that people understood how Basil distinguished situations according to touch relations.

Direction

Because the graphical world is spatially ordered, Metamouse infers directional constraints on movement and on the order of selecting objects. The bias includes five direction terms (up, down, left, right, any). Other paths can be expressed by construction (combining a basic direction with a touch constraint). When selecting objects in order, the region searched is the half-plane beyond the point at which the previous object was selected. The questionnaire [12] showed that people assumed search would occur in a cone rather than a half-plane, unless a line was shown to sweep across the region. Thus in our user manual we explained that search is like sweeping with a wide broom.

Object selection

An object is assigned to variable V in action A, if it meets two conditions: first, it must satisfy all touch relations in A that include V; and second, it must satisfy the "selector function" associated with V, which tests usage criteria such as "the object has not been assigned to this variable previously," and ordering of selection, as in "the next object found when searching along direction (right)". Three selector functions distinguish between searching the display list versus creating or re-using an object. The selector is deduced from a single example (by checking for previous occurrences of the given object); this deduction may be incorrect, as when re-selecting an object to initiate a second iteration over the display list.

During the pilot study we tested a "doctrine" [3] that explained variables and selector functions, but users did not benefit from this information and complained it was too much to absorb, so we reverted to the one-page manual.

Flow of control

Metamouse is described as an apprentice, watching what the user does until he thinks he knows what to do next, whereupon he starts predicting. Sequential flow of control seems a natural way for users to demonstrate complex tasks. One bad aspect of our design is that selector functions influence flow control; when re-use is inferred instead of search, the system fails to form a loop. Users must organize a task so that it iterates only once over a set of objects. We did not state this in our manual, but should have.

2.3. Metamouse's use of examples

Metamouse's learning ability is by no means general, but is based on analyzing a single example as in explanation-based learning in a weak theory domain [14]. With two minor exceptions (concerning direction and input versus constant location) it does not generalize from multiple examples, but rather stores mismatches as alternatives, in the manner of exemplar-based learning [2]. This finesses the problem of deciding whether to generalize or create a conditional branch, at the cost of occasionally misclassifying a situation. Metamouse would tend to make more mistakes than a true exemplar-based system because it uses the first match obtained by testing in recency order, rather than the best match according to a similarity measure.

2.4. Recent changes to Metamouse

Inference in the absence of touch constraints

Earlier versions of the system prompted the user for an explanation whenever an action resulted in no touch constraints. The response options were that position or distance was a constant or input, or that the user forgot to make a construction. In the current version we replaced this with a more "elegant" approach. By default the system infers a constant distance moved (as certain commercial drawing programs do) but reminds the user that Basil likes constructions, and lets the user to adjust an object after it is moved. From multiple examples the system infers whether position or distance are constant or input.

A new algorithm for constructing procedures

Previously reported versions of Metamouse used a learning algorithm that learned a state machine which, by virtue of its manner of construction, was equivalent to production rules whose context comprises one previous action. Since fixed length context cannot capture state, contexts were often too general so that Basil made bad predictions, for instance whenever the first iteration of a loop differs from the rest, which happens often. On the other hand, using a large context would preclude many good predictions. We therefore installed a new algorithm that specializes a rule's context only when the user rejects its prediction. The restricted context must cover all previous firings of the rule; otherwise, the algorithm extends the context of the user's corrective action instead. When predicting, contexts are tested in length-order, so that the most specific is tried first.

Promoting constraints based on object re-use

One weakness inherent in identifying constraints from a single action according to a weak domain theory is that a touch ignored as coincidental may introduce an object for later use. The first implementation of Metamouse failed to re-use such objects. To remedy this, the new rule for inferring re-use checks coincidental touches, and reclassifies them as constraints if they introduce the object of interest.

Checking for future conflicts when binding variables

It may happen that some variable V in the current action

becomes bound to object Q, and that some future step includes relation touch(V : W), where W is bound to Q already. This problem may sound arcane but can occur whenever a tool of type T is used on a set of objects of type T. For instance, in our pilot study (prior to this change), one subject used a box to separate other boxes. Inevitably, Basil chose that tool as the next box to edit (which the user accepted), then tried to move it to the opposite side of itself, whereupon the program promptly crashed.

When binding a variable, the constraint solver now checks all references to it in reachable future actions for possible conflict. The solver checks current bindings only, so if the conflicting variable is in fact rebound by some intervening action our solver will nonetheless reject the proposed value. Although this conservative approach will fail to select some valid objects, the alternative is to anticipate entire future computations, which seems infeasible.

3. Experimental design

The goal of the user study was to discover whether people could put Metamouse's programming-by-example to effective use in practice, and to identify shortcomings in its design and implementation.

We collected both quantitative and qualitative data. Quantitative data were: first, time to perform tasks with and without Basil's help, to determine whether Metamouse improves user productivity; second, scores on a post-test of user's understanding of Basil's behavior, to assess the ease with which the metaphor is learned; and third, ratings on an "ease-of-use" questionnaire. Qualitative data were: first, video traces of what people actually did to solve editing problems with Metamouse; second, user protocols while at work, indicating their insights into how to use the system; and third, user comments (during and after the session) regarding usability and specific problems they encountered.

In the event, we found the qualitative data most useful, and opted to end the trials after 4 pilot and 3 experimental subjects, since we found that the anecdotal evidence for the need to upgrade Metamouse was quite overwhelming by then.

3.1. Hypotheses and questions

Regarding object selection

We expected that users who did not initially decompose tasks in a way compatible with Basil's selector functions would not learn to do so. The results confirmed this hypothesis; moreover, nearly every subject used the incompatible decomposition.

Regarding the use of construction

We expected that users would create appropriate construction tools after reading the one-page manual and observing that Basil does not perform correctly without them. The results indicate that this hypothesis should be refined; users readily adopted "static," declarative constructions for spatial relations but had no insight into using "dynamic," procedural ones like a sweepline for expressing intrinsic properties (such as height). Moreover, users invented constructions for alignment, but had to think hard before inventing a tool for spacing.

Regarding directional scanning

Can users understand and exploit Basil's notion of scanning along an axis for the next object? We found that users would process a set of objects in order along a major axis, but often they would begin with an item in the middle of the set. They were not consciously teaching Basil to scan across the set. Moreover they did not transfer the notion of scanning to other applications like sorting by height.

Regarding inference of intrinsic properties

Would users expect Basil to infer intrinsic properties like object height? None of the pilot or experimental subjects believed Basil knew about height. All were puzzled when asked to teach him how to sort by height, because they believed they were supposed to find some way of expressing height in terms of touch. Several users said they would prefer a more direct way of instructing Basil with commands.

Regarding eager prediction

Would users prefer that Basil wait for several examples? None of the users complained about Basil's eagerness, and there were surprisingly few complaints about his mistakes. We suppose this was because they were more concerned with the trouble they had inventing suitable constructions. Most complaints concerned Basil's inability to learn from the techniques that users actually tried.

3.2. Conditions

We designed a quasi-controlled experiment [5]. The experimental conditions were as follows. All subjects read the same user manual and performed the same tasks in the same order on identical data. Subjects were chosen non-randomly from three distinct professional groups. Each task was performed with and without Basil. For the quantitative part of the study we intended that use of Basil would be the independent variable, task performance time the dependent.

As noted earlier, the "user manual" for Metamouse is a one-page informal description of Basil. In the pilot we tested a complete Metamouse doctrine as well, with a view to making the amount of instruction an experimental variable. The pilot subjects, however, found the doctrine too deep, so we decided to test with a minimum of explanation.

A facilitator (who was not familiar with the system's internal workings) was on hand to help subjects learn the tasks and how to operate the drawing program. The facilitator did not advise subjects on how to teach Basil.

3.3. Tasks

We asked our users to perform six tasks at least three times each: first, without Basil's help, then with Basil, and finally without him again. We measured the time to complete each run. The times for the second and third runs were compared, so that practice effects would bias against Metamouse rather than for it.

Users were allowed more than one practice run without Basil, so that they would understand the task, and more than one trial with Basil—perforce because the program often crashed!

The first two tasks were quite simple and solvable without construction tools, so that users would get a feel for working with a predictive interface. The middle two were rather difficult, since they involved spacing and sorting. The last two were easier and were based on alignment.

Task 1: Squeeze and align

In Task 1, the user edits three clusters of three boxes each, as shown in Figure 1. The user is supposed to squeeze the middle box and drag the rightmost one till all three are aligned along their bottom edge.

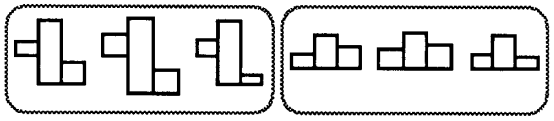

Figure 1. Before (left) and after Task 1.

No constructions are required for this task. Under ideal conditions Basil can predict all edits to the second and third clusters after the user repeats a selection.

Task 2: Pancaker

This is a variant of the previous task, in which the third box must be moved up and then squeezed so that all three boxes are the same height, as shown in Figure 2. To illustrate the "generality" of Basil's constraint inference, the third box must be "unsqueezed" since it is too thin.

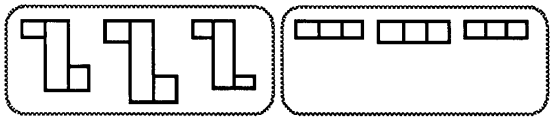

Figure 2. Before (left) and after Task 2.

No constructions are required for this task, either. Under ideal conditions Basil can predict all edits to the second and third clusters.

Task 3: Picket fence

The user moves six boxes so that they are spaced evenly and remain aligned at the bottom, as shown in Figure 3.

For this task Basil needs a construction for spacing. Ideally Basil could predict edits on the last four or five boxes (depending on whether or not the first is left where it is).

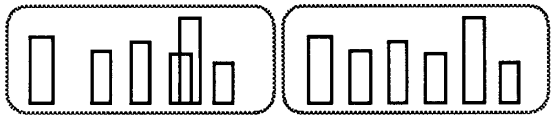

Figure 3. Before (left) and after Task 3.

Task 4: Sort by height

The user sorts six boxes so that they increase in height from left to right, as shown in Figure 4. The boxes must remain aligned and evenly spaced, as in Task 3.

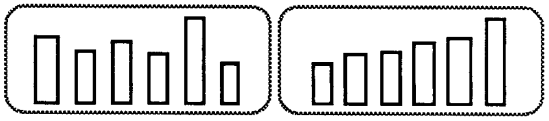

Figure 4. Before (left) and after Task 4.

In addition to a construction for spacing, this task requires that the user select objects in height order (either direction). Two examples may be required before Basil generalizes the heading to upwards (eg. moving to the tallest box could be mostly rightwards, and to the next tallest, leftwards). Ideally, Basil could sort five of the six boxes.

Constructions and performance for the spacing/alignment subtask are as for Task 3, but for Basil to predict these actions, they must be interleaved with sorting, since the system cannot infer two loops over the same set of objects.

Task 5: Stairway

The user is given five boxes scattered about the display, to be aligned at their lower-right corners along some arbitrary axis, as shown in Figure 5.

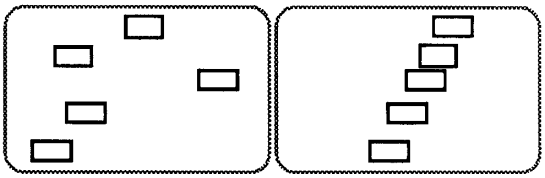

Figure 5. Before (left) and after Task 5.

This task requires a line construction for the alignment axis. Boxes can be selected in any order, but if the user skips a box, Basil will not predict it until some user action has caused him to generalize the search path to "any direction."

Task 6: The dreaded org chart

Given a set of boxes all connected by tie-lines to one other box, the user is supposed to move them into alignment at the right, and reconnect the tie-lines, as shown in Figure 6.

A guideline is used as in Task 5. For Basil to predict editing the tie-lines, the user's procedure must form a single loop (drag-box, drag-tie-line)*, as in Task 4.

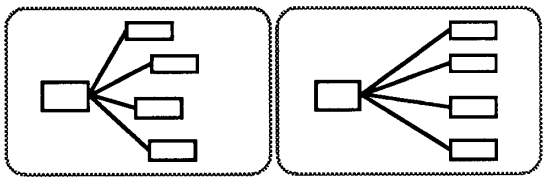

Figure 6. Before (left) and after Task 6.

3.4. Subjects

We wanted a pool of subjects that represented different classes of potential Metamouse users: drafting professionals, who we presumed would find constructions quite natural; graphic designers, who would be accustomed to a very subtle geometry measured by eye; non-graphic computer users (like secretaries) for whom the techniques would be completely novel; and computer programmers, who would have algorithmic insights.

In the event, we performed a small study with one graphic designer, three programmers and three geologists. The geologists enabled a stringent trial, since they had little experience with drawing programs and their years of work in contour mapping biased them strongly towards using visual inspection as opposed to constructions. They had no trouble aligning and spacing objects almost perfectly by eye.

4. Observations

We collected task timings, video traces, user profile questionnaires, post-session knowledge tests and user evaluations of the system. Space permits (and sample size justifies) reporting only the strongest trends in these data.

4.1. Data

The timings

Due to program crashes, complete task timings were obtained with only five of the seven subjects for tasks 1, 2, and 5. Tasks 1 and 2 took longer with Basil (an average of 56 seconds) than without (44 seconds). This result can be discounted for two reasons. First, subjects had not yet become accustomed to Basil's prompts and sometimes waited for him to perform the next action rather than respond to the previous one. Second, the facilitator neglected to emphasize that objects should be aligned precisely, hence users did rather slapdash work when performing the tasks by themselves. In contrast, they were quite deliberate when teaching Basil, placing objects precisely, despite being told that Basil infers exact vertex-to-vertex touch from near misses.

Task 5 took an average of 74 seconds with Basil as opposed to 66 seconds without. In this case there are no excuses. When doing the task manually, most subjects aligned by eye and thereby saved time at the expense of some precision. When teaching, several people used constructions that reduced Basil's ability to predict. The time they spent creating these and pondering his failures reduced productivity.

Despite the timings, subjects reported in the user evaluation that these same tasks were accomplished faster using Basil. This may be due to the fact that Basil does single actions instantaneously on a SPARCstation 2.

The user comprehension post-test

After completing the tasks, each user was asked eight questions about the operation of Metamouse, which could be answered true, false or don't know. Three questions concerned basic facts about the interface, such as whether moving the cursor onto a tack causes the objects it connects to be highlighted (true). The other five cover the knowledge of Basil's bias that users would need to understand his behavior. Table 1 shows how users' beliefs matched the facts.

This table provides some insight into the problems that users encountered with Basil. The manual convinced them that spatial relations other than touch are ignored, but they have not realized that touch can be used to identify structure, and they are unsure whether intrinsic properties are sensed. The users were also unclear about the status of program bugs after they made corrections.

Question states that...	Correct belief	Incorrect belief	No opinion
Relations other than touch are ignored	6	1	—
Intrinsic properties (e.g. size) are ignored	2	3	2
Structured objects are recognized by touch	2	—	5
Rejected actions are retained at low priority	2	3	2
The larger component of a direction vector is chosen	4	1	2
Totals	16	8	11

Table 1. Count of users true/false beliefs about Basil.

The questionnaire

The subjects answered a dozen questions on the usability of Metamouse. All considered it helpful on tasks 1, 2, 5 and 6, and a definite hindrance on 3 and 4, which they found difficult or impossible to teach. Subjects were split on the issue of eager prediction; about half of them would have preferred to invite Basil to start predicting.

Tasks 1 & 2

The first two tasks, involving no constructions, presented no difficulty to the users. Although they had no prior exposure to the interface, all but one responded without help from the facilitator to Basil's prompt for approval of his first prediction, apparently because he squeaks. Basil does not squeak on subsequent predictions; some users waited several seconds for him to continue before clicking "OK".

Task 3

When doing Task 3 without Basil's help, all subjects estimated the spacing by eye. On their first attempt to teach the turtle, two of the pilot subjects (programmers) used a box as a spacer and a line drawn through the base of all boxes to keep them level. This second tool precluded the system's

making any predictions. One of the geologists also anticipated the need for construction, but remarked "If you're going to have to make him a tool to move it, it's just as easy to move it yourself." After three attempts to teach Basil without one, she announced "Oh, I could draw a line!" and drew a spacer. Unlike the programmers, she recognized that it would maintain alignment.

Of the remaining four users, two indicated after a failed attempt to teach Basil that they would need some sort of tool. After long pauses to think, one of them drew a line through all boxes, as shown in Figure 7, and was disappointed that Basil did not start predicting. Apparently the user was able to express his mental focus on a horizontal relationship, though unable to specify it.

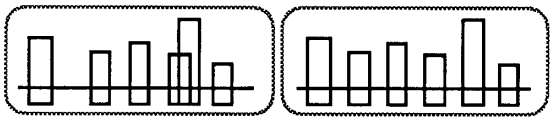

Figure 7. Tool intended to focus on horizontal spacing.

Task 4

When doing task 4 without Basil, subjects compared heights by eye, though one moved some boxes next to one another when doing so.

Users anticipated that this task would be hard to teach, and their methods when demonstrating for Basil were more complicated than when doing the task for themselves. Ironically, though all had selected boxes in order by height when doing the task manually, most adopted complicated shuffling procedures (sometimes resembling bubblesort) when trying to teach the turtle.

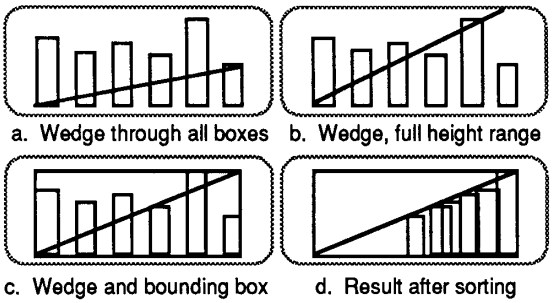

Figure 8. Tools attempted for sorting by height.

One of the non-programmers, upon repeating task 4 after doing 5 and 6, developed a wedge tool for sorting, shown in Figure 8. No version of his tool would have enabled Basil to predict the sort (due to inadequate selector functions), but the user's attempts to build it are interesting. The first version (Figure 8a) clearly couldn't work because it does not cover the range of box heights. The user fixed this (8b) and then adduced Basil's failure to predict to the fact that the line touched some boxes but not others. To correct this the user drew a bounding box around all the rectangles (8c) and was disappointed that Basil still failed to predict. This final version also expressed the constraint (necessary for correctness) that boxes move horizontally.

The user who initially rejected tool use in task 3 did the same in task 4, remarking that "What you need is a program you can tell to sort," and did not discover tools for this task.

One user made a baseline on his third attempt to teach a sort. Recalling that only touch matters, he made sure when shuffling boxes that their temporary "holding" positions lay directly underneath and touching other boxes.

Task 5

When doing the task manually, most users aligned the boxes by eye. One of the geologists drew a guideline at the start, another aligned the first three boxes by eye and then drew a line through them to help her with the last two, and the third made a guideline after completing the task in order to check his work!

When teaching Basil, six of seven users drew diagonal guidelines, and indeed without hesitation on their first attempt. Subjects preferred to draw their guideline so that it touched at least one box (often the rightmost), rather than draw in empty space. One person moved two boxes into alignment and then drew the guideline along the diagonal that they formed. Another made a tool out of two lines leading upwards and downwards from the corner of the rightmost box. All these reasonable tactics reduced the number of predictions Basil could make.

One person used a sweepline to constrain boxes vertically. Another drew horizontal lines from each box to the guideline in order to constrain the boxes to their vertical position: in principle Basil should be able to learn this iteration over boxes, but due to a bug the "draw-line" steps were not matched and therefore no loop was predicted.

Task 6

When practicing task 6, one user made a guideline for himself; the rest aligned by eye. As in the previous task, all users but one drew a guideline when teaching Basil, and made it touch at least one box. One user moved the first box, then drew a guideline through it.

Most users picked boxes in top to bottom order, except for their first selection, typically the rightmost box. The one who used a sweepline in task 5 did so again. Five of seven users processed all boxes first, then all lines; a task decomposition Basil cannot learn. Thinking aloud, one geologist tried to explain the lack of predictions: "I probably wasn't supposed to move the lines that way... I wonder if it's confused because of the boxes rather than the lines." The graphic designer guessed that her procedure was to blame and then used the correct (move-box, move-line)* method.

5. Discussion

5.1. Construction tool intuition

Nearly all users recognized the need for construction tools. With few exceptions, users anticipated the need to construct

more complex relations such as arbitrary alignment; for example, they were generally quite proficient in their use of alignment tools for tasks 5 and 6. Since we did not vary the order of tasks, we cannot say whether this was due to practice. The only evidence of practice effects is that the subject who returned to task 4 after 5 and 6 was able to develop a suitable construction.

Programmers used more and better tools in tasks 3 and 4, but geologists used simpler, more effective tools in tasks 5 and 6. Generally, non-programmers gave simpler and better explanations of tool failure; programmers tried to second-guess the implementation (hopeless speculation!).

Dynamic vs static tools

Only one user (a programmer) created a moving tool (a sweepline); all other tools expressed static spatial relationships. Several subjects complained that the instructions regarding dynamic tools were impossible to understand. We should like to further investigate differences between the use of dynamic, procedural tools and static, declarative ones.

Order and direction

Our study shows that users do not process a set of objects in perfect order along some dimension, yet assume nonetheless that Basil will iterate in order over the entire set.

Alignment vs spacing

Users seemed to have better intuitions regarding constructions for alignment than for spacing. For example, in task 3, one person tried to use a line through all boxes as a spacing tool (Figure 7). While people could explain to themselves why Basil failed to notice a spatial relation, most (especially programmers) expected him to observe some simple relations like constant spacing or alignment along a vertical or horizontal axis.

Intrinsic properties

Only one subject discovered a suitable tool (a declarative one at that) for sorting boxes by height. Nearly all subjects believed from the outset that Basil would not infer the selection of boxes in order by height, even though a minority were certain that he does not observe such properties. We agree with the user who suggested that more direct access to properties would improve instruction.

5.2. Object selection

Although Basil's users did learn quickly how to make some useful constructions, they were punished for their efforts. Often, though not always (which makes the problem worse), Basil could not include previously encountered objects in an iteration. Set iteration in Metamouse is inferred when a user action matches an existing action including the "find-novel-object" selector function. If the user intends to form a loop over objects previously encountered, the "reuse-object" selector will be inferred instead. Once a loop is formed, however, objects assigned previously to other variables can be selected by find-novel-object. Thus if a guideline touches one or two boxes, Basil can iterate over the entire set, including those two, once the user has selected some box not encountered before. But a baseline drawn through all objects, and ostensibly defining them as a group, precludes iteration.

This problem has two other symptoms prevalent in our user study. First, Basil cannot learn two iterations over the same set of objects. The task decomposition users preferred for task 4, (sort-boxes)* (space-boxes)*, was unlearnable. Second, Basil cannot learn to iterate separately over each dimension of a set of pairs of objects that touch one another, like the (box, tie-line) pairs in task 6. Two ways of decomposing edits over sets of structured objects were predicted in [22]: iterate over members of the set (editing all substructures of one member before doing the next); or iterate over each dimension of the set (process all occurrences of each substructure in turn). Metamouse learns the first decomposition only; with few exceptions, the users in our study tried to teach the second.

5.3. One-shot learning

Our study reveals that although users expect the system to learn from one example, Metamouse's inability to generalize actions from multiple examples is a serious weakness. Analyzing one example's features is indeed useful for choosing preferred features to match with subsequent examples, but lacking the ability to update a generalization based on further evidence, or choose the best match from several exemplars, the system makes too many bad predictions and is incapable of learning to discriminate situations that look the same when analyzed in isolation.

5.4. Improving Inference in Metamouse

The vast majority of Basil's failures to predict in this study, and the attendent frustration and confusion for subjects, can be traced to the selector function find-novel-object. We propose two extensions to the system to remove this blockade. First, selector functions should represent ways of grouping objects into sets. Second, the system should be able to retract one hypothesis and try another.

We propose selector functions for the following kinds of sets: all objects (of a given graphic type) in the display list (eg. all boxes); all objects touched by the same object simultaneously (eg. all boxes on a baseline); and all objects assigned to a given variable throughout its history (eg. all tie-lines encountered while iterating over a set of boxes). Selection may be in spatial or temporal order.

When matching two actions, the system will choose a selector that applies to both. If Basil fails to predict or chooses the wrong object, the system will compare the offending program step with the user's corrective action and if possible match them by altering the selector function.

Once we have implemented these changes we will test a new group of users on the same tasks. Should the system pass these trials, a study using more complex tasks would be warranted.

6. Concluding remarks

The results of the user study indicate (a) that we have over-committed to the procedural approach, (b) that more spatial relations should be inferred without the need for construction, (c) that some means-end analysis of constructions may (unfortunately!) be necessary, and (d) that sets of objects should be represented explicitly and inferred from spatial relations as well as iteration histories.

Finally, a word about implementation. Metamouse is a large and complex program. The current version consists of 11k lines of documented C++. The core of the learning system comprises 1k lines, but its interface to the application includes a constraint solver (2k lines) and code for accessing state and history (3k lines). The combination of interactive graphics, constraint manipulation, and inference makes it an intricate and difficult program to work with. We observed with regret that our users encountered many new bugs in the system, which supposedly had been thoroughly tested. The fact is that the metaphor encourages users to behave creatively by discovering novel ways to teach and novel constructions to "explain" aspects of what is taught, and the system applies inferential methods to make, and execute, generalizations about the behavior.

Constructing and debugging systems of this nature is intrinsically difficult. This justifies extensive testing of simulated systems before implementation.

Acknowledgements

This research is supported by Apple Computer Inc. and by the Natural Sciences and Engineering Research Council of Canada. Valerio Franceschin and David Astels wrote the code. Donna Choquette ran the experiment.

References

1. P.M. Andreae. "Justified generalization: acquiring procedures from examples." PhD. Dept of EE & CS, MIT. 1985.
2. R. Bareiss. *Exemplar-based knowledge acquisition.* Academic Press. San Diego CA. 1989.
3. B. Bell, J. Reiman, C. Lewis. "Usability testing of a graphical programming system: things we missed in a programming walkthrough." *Proc CHI '91.* May 1991.
4. A. Borning. "Defining constraints graphically." *Proc. CHI '86.* April 1986.
5. L. Cohen, L. Manion. *Research Methods in Education,* 3rd ed. Routledge. London. 1989.
6. A. Cypher. "EAGER: programming repetitive tasks by example." *Proc. CHI '91,* pp. 33–40. May 1991.
7. D.C. Halbert. "Programming by example." Research report OSD-T8402. Xerox PARC. Palo Alto. 1984.
8. R. Heise. "Demonstration instead of programming: focussing attention in robot task acquisition." MSc thesis. Dept. of CS, U. of Calgary. 1989.
9 H. Lieberman. "An example based environment for beginning programmers." *Artificial Intelligence and Education,* pp. 135-151. Ablex. Norwood NJ. 1987.
10. B.A. MacDonald, I.H. Witten. "Programming computer controlled systems by non-experts." *Proc. SMC.* 1987.
11. D.L. Maulsby, K.A. Kittlitz, I.H. Witten. "Metamouse: specifying graphical procedures by example." *Proc. SIGGRAPH '89,* pp. 127-136. August 1989.
12. D.L. Maulsby, G.A. James, I.H. Witten. "Evaluating interaction in knowledge acquisition." *Proc. EKAW '89,* pp. 406-419. July 1989.
13. D.L. Maulsby, I.H. Witten, K.A. Kittlitz, V.G. Franceschin. "Inferrring graphical procedures: the compleat Metamouse." *Human Computer Interaction* (in press).
14. S. Minton,, J.G. Carbonell, *et al.* "Explanation-based learning: a problem solving perspective," in J.G. Carbonell, ed., *Machine Learning: Paradigms and Methods,* pp. 63-118. MIT Press. Cambridge MA. 1990.
15. T.M. Mitchell "The need for biases in learning generalizations," in J.W. Shavlik, T.G. Dieterrich, eds., *Readings in machine learning,* pp. 184-191. Morgan Kaufmann. Palo Alto CA. 1990.
16. Mo Dan Hua. "Text editing procedures from examples." MSc thesis. Dept. of CS, U. of Calgary. 1989.
17. B.A. Myers. *Creating User Interfaces by Demonstration.* Academic Press. San Diego. 1988.
18. B.A. Myers. "Text formatting by demonstration." *Proc. CHI '91.* May 1991.
19. R. Nix. "Editing by example." PhD. Computer Science Department, Yale University. 1983.
20. F.P. Preparata, M.I. Shamos. *Computational Geometry.* Springer-Verlag. New York. 1985.
21. J.T. Stasko. "Using direct manipulation to build algorithm animations by demonstration." *Proc.CHI 91.* May 1991.
22. P. van Sommers. *Drawing and Cognition.* Cambridge Univ. Press. Cambridge UK. 1984.