We describe how a computer system can acquire procedural knowledge from end-users in the domain of interactive graphics (Maulsby & Witten, 1988). This domain provides excellent examples of human-computer cooperation requiring continual, richly-structured, interchange between the partners. Popular drawing programs (eg MacDraw; Cutter, 1987), which augment the software analog of drafting tools with extensive editing capabilities and rudimentary positional constraints, are an ideal environment in which to investigate the acquisition of knowledge from non-expert computer users. These facilities enable casual users to master easily some of the mechanical skills of draftsmen and to concentrate more upon design. Users often perform individual operations in fixed sequences which, in principle, constitute knowledge that may be re-used later. Current drawing systems do not provide any way to manipulate this knowledge, and in practice users suffer considerable frustration because they have to repeat operations over and over again.

As an example, Figures 1–3 illustrate several tasks to be programmed within a MacDraw-style utility. The input picture is transformed into the output picture by constructive operations having both *ad hoc* and derived parameters. Most users can, after some contemplation, generate suitable sequences of operations to accomplish these tasks. They do so by operationalizing and combining knowledge from diverse sources, ranging from spatial intuition through principles of Euclidean geometry to an understanding of everyday routines such as sorting. Our job is to devise a mechanism to acquire these procedures, capturing them in a form which is flexible enough for them to be re-used later in slightly different circumstances. It may be possible to analyze the procedural knowledge so acquired and abstract more general principles from it, but we have not yet begun to attempt this.

We view knowledge acquisition in this domain as tantamount to *teaching*. Unlike other researchers in machine learning, we stress the transitivity of the verb ‘‘teach’’ — there must be some organism that is being taught, and the teacher must be aware of its nature. A cornerstone of our approach is to employ a suitable metaphor through which users can conceptualize the target of the teaching process. A user who has a clear idea of the capabilities of the learner will automatically adapt his exposition to capitalize on its strengths and compensate for its weaknesses. The success of the metaphor will determine how well the user's teaching matches the student's ability to learn.

Metamouse is the metaphor.

Welcome to my world

by M. Mouse

I, Metamouse, live in Flatland (Abbott, 1884), sharing this world with just two types of objects, lines and boxes, on which two kinds of operations are possible, translation and scaling. These primitives form the basis of my drawing world, a program called A.Sq, whose user interface methods resemble those of MacDraw. Although computer drawing environments normally provide a much richer selection of primitive objects and operations, I am an experimental animal and this limited domain is rich enough for experimental purposes.

My teacher, the human user of A.Sq, can create objects by rubber-banding, select one of them with a pick, and transform a selected object by moving its handles. For example, lines can be translated, rotated, elongated or shortened. Boxes can be translated, reshaped, stretched or shrunk. Programs for this world which the user teaches me are composed of these same operations.

As far as I am concerned, an object is characterized by its *type* (line or box) and its *parts*. The parts include the line segments that comprise it, and certain distinguished points of the object, called its *handles*. A line has just one segment, while a box has four (*top*, *left*, *right*, *bottom*). As Figure 4 illustrates, a line has three handles (*point1*, *mid*, *point2*), while a box has nine (among them *topleft*, *topmid*, *center*, *bottomright*). For the user's convenience, I label objects as he sees them on the screen, so that ''left'' means *his* left, not necessarily mine.

In my world, A.Sq, a particular drawing function is associated with each handle of an object. A line can be translated by grabbing its midpoint and moving it; it can be rotated, elongated, or shortened by grabbing either end and moving that while the other end stays fixed. Boxes can be reshaped by moving any corner, stretched or shrunk by moving the midpoint of a side. I do not know anything about the geometric properties of such transforms — I just grab handles and move as I am told.

I am also concerned with relations between objects. There are three kinds of relation: *point–on–point*, *point–on–segment*, and *segment–meets–segment*. Each of them is symmetric. The first occurs when handles of two objects coincide in space. For example, if *L* is a line and *B* is a box, *L.point1–on–B.topleft* is a point–on–point relation. The second kind occurs when a handle of one object lies on a segment of another, like *L.point2–on–B.bottom*. The third kind is when a segment of one object intersects that of another, like *B.right–meets–L.lineseg*. The point of intersection is not specially distinguished — I do not know where objects meet, just *that* they meet.

Conventional computer-graphics ''gravity'' applies throughout my world, so that points are deemed to be coincident if they are close enough, and if this is the case they fall automatically into a position where they really are coincident.

This is my body

by M. Mouse

I mimic the behavior of the standard mouse locator device. Although my name is a mouse's, my body, in deference to Papert's creations, is a turtle's (Papert, 1980; Abelson & diSessa, 1980). People call me a moveable menu that sometimes runs away. My basic operations are to move forward, back, left or right, and to grasp or let go. To make me do this, the teacher clicks the ordinary cursor (I call it the ''dumb'' cursor) on the appropriate part of me as shown in Figure 5. When I think I know what to do, I do it anyway without waiting for the teacher; this is why I sometimes over-enthusiastically "run away". If the teacher does not approve of a particular action, he or she can always pull me back by the tail.

In order to transform an object, I grasp it and carry it to its destination. Note that I, like my teacher, can only grasp objects by their handles. To facilitate the programming of problems that involve rotational order (eg convex hull), I can be instructed to perform true rotation. To rotate an object, I grasp one of its handles and turn myself. I can only grasp one thing at a time. Sometimes I grasp things to move them around. Other times I pick up a constructed object which functions as a tool, perhaps a short line to use as a spacer, or a long one to use as a sweep line — like a pair of sensitive whiskers.

My sensory feedback comprises current position, heading, and tactile events, the last being by far the most important. I use position only as a constant, when my teacher specifies "same-place" generalization. I refer to my heading only when I have been rotating and must return to some previous orientation. My tactile state is governed by what I am grasping and also by what I am touching with my snout. I can only grasp one object at a time, but may touch several objects at once. Although I am very short-sighted I have a well-developed sense of touch, for I can tell

- the type of each object I am touching
- the type of the tool (or object) I am grasping
- the specific part of each object I am touching
- the specific part of the tool (or object) I am grasping
- spatial relations between the tool (or object) I grasp and any other object it touches.

The parts of an object are its handles and line segments, while spatial relations are the three binary symmetric relations I described before.

Note carefully the limits placed on my tactile awareness. I only pay attention to things I bump into, and my head flashes when this happens so that my teacher knows of the event. I sense only at my snout (current exact position). I can sense the objects I touch, and if I am holding a tool I can sense the objects it touches. However, I am not aware of contacts between other objects and what I am touching, nor of contacts between other objects and those that touch the tool I am grasping. This extra information is not strictly necessary since I can always check for significant contacts by moving (or being moved) to the location where they might occur. My limited sense of touch is designed to prevent my teacher from believing that I can analyze the entire display.

All I know

by M. Mouse

I learn programs, which are structured sequences of actions (more on that later). Often the actions of a program are parametrized, and some of the parameters must be variable for the program to be general. The program must be deterministic so that it is clear to me which action to perform when branching occurs. I therefore model actions as (precondition, operation, path, postcondition) tuples, in the manner of the STRIPS planning system (Fikes & Nilsson, 1971). The *operation* part must contain one of the basic operations I can do. The *path* part specifies my direction of motion. *Conditions* are generalizations of my sensory feedback, that is, logical combinations of spatial relations that must hold for the condition to succeed.

For example, here is a program step that I might learn. *If* I am facing right and grasping a box by its translation handle, *then* move forward *until* the bottom-right handle of the thing I am carrying (the box) touches the left side of any other box. In order to carry out this action, I perform an A.Sq *translate* command whose numerical parameters are calculated from the distance a ray would travel from my snout along the specified path before intersecting a ''left edge''.

I execute a program step provided that 1) its preconditions are a generalization of the current sensory state, and 2) its post-conditions are attainable. To determine if the post-conditions are attainable I imagine carrying out the action and see if it is possible to achieve the spatial relations necessary for the condition to hold. If I decide against executing a particular step, I check any alternative ones. If none remain, I ask my teacher to show me what to do in this novel situation. If he demonstrates one of the actions I have considered but rejected, I generalize its preconditions to accommodate the current situation by adding it n as disjunction. If the action is new, I create a new branch using the current situation as a condition.

Whenever my teacher adds a new program step, I check whether a ''sufficiently similar'' step appears elsewhere in the partially-constructed program. I look for a step with matching action, pre-condition, and post-condition. If I find one, I conjecture a link from the previous step to that one, instead of creating a brand new program step. This usually allows me to predict a new action, which I perform immediately, and if the teacher does not reject that action (by pulling me back) I take the conjecture as correct. At present, once a linkage has been confirmed in this way I never retract it. If my teacher rejects subsequent predictions, I try to construct a conditional branch around the rejected actions. (It is recognized that this deterministic model will likely prove too restrictive for teachers, and it may be necessary to adopt limited backtracking in the future.)

I strive to follow good programming practice by appropriately biasing my search for linkages. I do this by searching outwards from the current program step to try to keep all linkages as local as possible. This permits a simple, reliable algorithm to construct programs from traces. It will be interesting to see if this preference for local connections tends to produce well-structured drawing programs of the kind that van Sommers (1984) found in his study of the procedures people use when drawing.

My working memory has four conceptual divisions. Long-term memory contains the procedure as learned so far; from this I conjecture linkages and predict actions. My medium-term state records what objects have been created during the current trace. Thus I recognize a constructor, such a sweep line, when I re-encounter it, and can be taught to return to that particular constructor. Short-term memory records the identities of objects most recently touched or grasped; this permits letting go of something in order to investigate it further by ''sniffing'' around its perimeter. My immediate memory contains the current sensory feedback.

I resolve any conflict between tactile feedback and position, heading, or path, by generalizing. I prefer to generalize the last three, and am reluctant to generalize tactile relations. While I could infer generalizations of these (eg by climbing a hierarchy), and this might greatly accelerate learning, I am haunted by the specter of over-generalization and the difficulty of correcting it. Moreover, daring, successful generalizations may delude the teacher into over-estimating my intelligence — for I am a very simple mouse — and failures would result in a loss of confidence. In my present form, therefore, I only generalize touch relations in the most minimal sense, by merely recording the disjunction of those that occur.

In future I may experiment with other generalization heuristics. In general, however, I would rather be simple and reliable than brilliant but capricious.

What I can do

by M. Mouse

Here is an example which shows what I can do. Consider the ''box-to-line'' procedure illustrated in Figure 6. The teaching process consists of leading me through a trace of the task. The teacher begins by showing me the input set by selecting the boxes. Then she places the guideline's two endpoints (Figure 6c). Observing the absence of a contact constraint, I classify the event as arbitrary and interrupt to ask through a dialogue box (illustrated in Figure 7) whether the location is constant or a run-time input. The teacher indicates that both points are to be specified at run-time.

I am then led through the main iterative sequence (Figure 6c-i). It is easy for me to observe that the objects transformed belong to the input set, and that iteration terminates when every member has been processed. In general, however, selection and iteration may depend on any number of properties of objects or situations. Therefore iteration must be conditioned and ordered on events that I can sense by touch. A horizontal sweep-line serves this purpose, and also constrains the boxes' path of translation. The teacher makes me draw the sweep-line near the bottom of the screen and indicates (through a dialogue box) that its initial placement is constant. She then makes me grab the sweep-line at its midpoint handle and moves me upwards, with the line, until it touches the bottom edge of some box (Figure 6d). The contact is the condition on which a box is selected for translation. When the line is swept past the last remaining box, I note that the sweep-until-contact action will fail, and this failure becomes my condition for terminating the loop. But I am getting ahead of myself.

Observe that the desired program applies the alignment constraint through goal-directed translation, where the goal is a visible contact. A suitable description of the goal, say "lower left corner of box in grasp is coincident with some point on guideline", is invariant over iteration on the input set. My job is to distinguish this contact event and induce its invariance.

When the sweep-line touches the first box, Teacher has me grasp the box and move it rightward until its lower right corner touches the guideline while its bottom edge remains on the sweep-line (Figure 6e). Teacher then moves me back to grasp the sweep-line and proceed to the next box (Figure 6f). When I am made to select this second box, the action patently repeats that of selecting the first one. Consequently I conjecture a loop and predict the translation that is to follow (Figure 6g). The second box, however, must be moved to the left. I am biased towards easily generalizing directions of movement, so this does not faze me; I leap out of the teacher's control and eagerly move the box on my own. Since she does not object, I have now learned the body of the loop, and operate on the next box by myself (Figure 6h-i).

After processing the third and final box, I recognize that I cannot complete the action of moving the sweep-line because the "contact" postcondition cannot be met. Hence I terminate the loop on the condition of being unable to perform its first step, and call upon the teacher to demonstrate what to do. At this point, she has me remove the sweep-line and the guideline (Figure 6j), and then announces that the lesson is over.

Concluding remarks

by Metamouse's creators

Now that Metamouse has described his capabilities, we would like to make some general remarks and draw some conclusions.

The user of a system for programming by example has much weaker assumptions about the system than a programmer working in a formal language. Its limited powers of induction are revealed gradually, and typically in disastrous occurrences that the user finds hard to interpret. It has been argued that the relationship between programmer and system is best understood as that of teacher and apprentice, and hence that the perspicuity of the teaching metaphor is vital to the success of the system (MacDonald & Witten, 1987).

The design of Metamouse has been strongly influenced by the results of an experimental investigation of user behavior in constructive graphical tasks, in which we studied the performance of a diverse group of people using MacDraw (no Metamouse). Each subject was given one hour to complete seven tasks; activity was recorded using a commercial system for programming by example (Affinity Microsystems, 1985). We concluded that Metamouse must clearly inform the teacher of his awareness, abilities, and limitations as a pupil. Metamouse promotes the illusion that he is paying attention to the teacher's activities by tracking the movement of the physical drawing tool (mouse or stylus), and also transmits feedback from the learning system (head flashing, dialogue boxes).

We have not yet completed a proper evaluation of how people interact with the new drawing tool. Nevertheless, we can already draw some conclusions from the work. First, user interaction can augment or replace domain knowledge in constraining the massive searches incurred by function induction without requiring that the user manipulate or even understand the internal representation. The three-way trade-off between search space, ease of teaching, and built-in knowledge can be readily investigated in the graphical domain. Second, an eager learner can reduce teacher noise and enforce felicity conditions on the teaching sequence (van Lehn, 1983). The learner's actions must be clearly visible, and it is of course vital that a convenient ''undo'' facility be provided to control its impetuosity.

Finally, while much machine learning research aims to improve the learner, and some addresses the problem of teaching, none has considered the teacher's image of the pupil. We believe this to be very important. It can be moulded by providing an actual device with given capabilities to serve as the focus of the teacher's efforts. Moreover, an appropriate metaphor — in our case, that of ''Flatland'' — can quickly and forcefully convey the limitations of the learner.

Acknowledgements

This research is supported by the Natural Sciences and Engineering Council of Canada. We gratefully acknowledge the key role Bruce MacDonald has played in helping us to develop these ideas, and the stimulating research environment provided by the Calgary Machine Learning Group. Metamouse in particular would like to thank Hermann Maurer for providing the opportunity, and the occasion, to express himself in this way.

References

Abbott, E.A. (1884) *Flatland — a romance of many dimensions.* Signet Classics edition, New York, NY.

Abelson, H. and diSessa, A. (1980) *Turtle geometry.* MIT Press, Cambridge, MA.

Affinity Microsystems (1985) *Tempo.* Boulder CO.

Cutter, M., Halpern, B., and Spiegel, J. (1985) *MacDraw.* Apple Computer Inc.

Fikes, R.E. and Nilsson, N.J. (1971) ''STRIPS -- a new approach to the application of theorem proving to problem solving'' *Artificial Intelligence, 2,* 189-208.

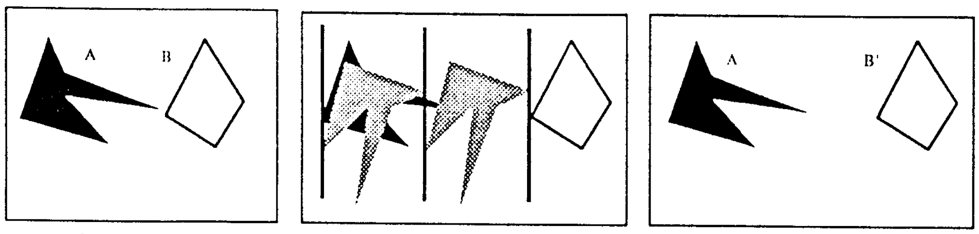
MacDonald, B.A. and Witten, I.H. (1987) ''Programming computer controlled systems by non-experts'' *Proc IEEE Systems, Man and Cybernetics Annual Conference,* Virginia, October 20-23.

Maulsby, D.L. and Witten, I.H. (1988) ''Acquiring graphical know-how: an apprenticeship model'' *Proc European Knowledge Acquisition Workshop 88,* Bonn, West Germany, June 1988; also available as Research Report 88/302/14, Department of Computer Science, University of Calgary, Calgary, AL.

Papert, S. (1980) *Mindstorms.* Basic Books, New York, NY.

Van Sommers, P. (1984) *Drawing and cognition.* Cambridge University Press, Cambridge, England.

VanLehn, K. (1983) ''Felicity conditions for human skill acquisition: validating an AI-based theory'' Research Report CIS-21, Xerox PARC, Palo Alto, CA, November.

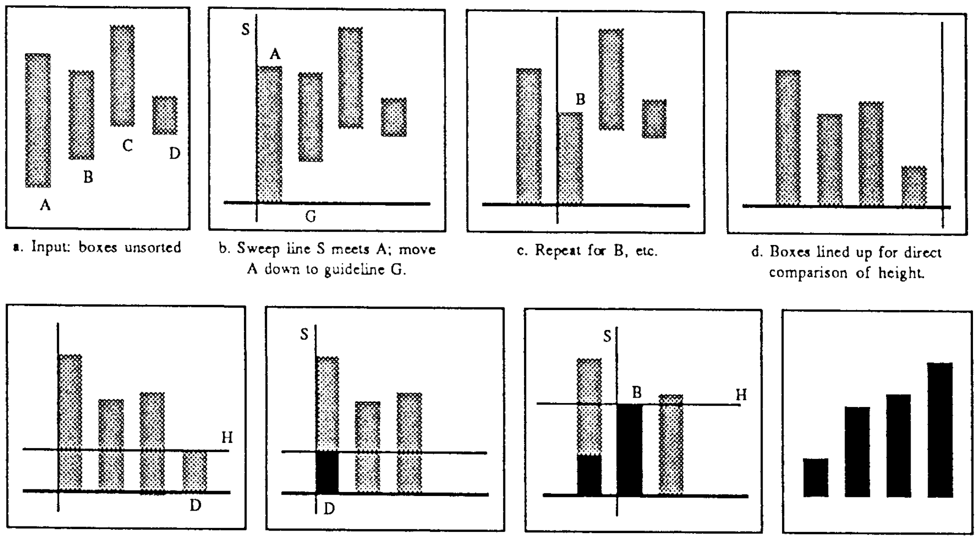


a. Input: two polygons

b. Construction: measure height of A by duplicating and rotating 90°; translate B.

c. Output: A and B' (=B transformed)

Figure 1. Graphical task: translate polygon B such that distance from left extreme of A to left extreme of B is 2 x height of A.



a. Input: boxes unsorted

b. Sweep line S meets A; move A down to guideline G.

c. Repeat for B, etc.

d. Boxes lined up for direct comparison of height.

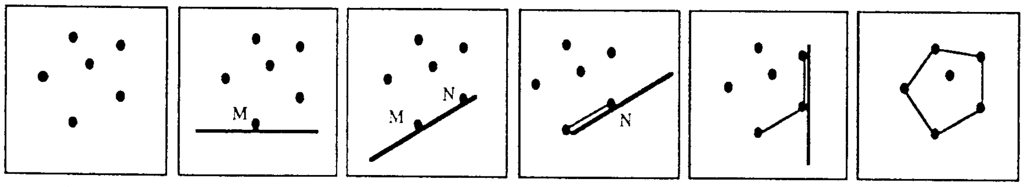
e. Create height-sweep line H and move until contact with top of some (lowest) box.

f. Move D to lateral sweep line S.

g. Advance S by const. distance; raise H to next contact; move B to S (null op.)

h. Output: boxes sorted by height and evenly spaced.

Figure 2. Task: given a set of boxes, sort them in order of increasing height.



a. Input set

b. Bring sweep-line up to lowest member, M

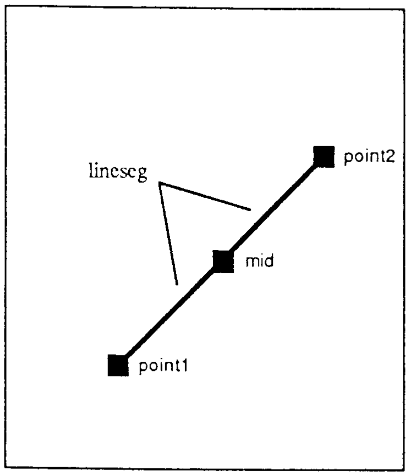
c. Rotate sweep line about M to contact N

d. Connect M, N; center sweep at N

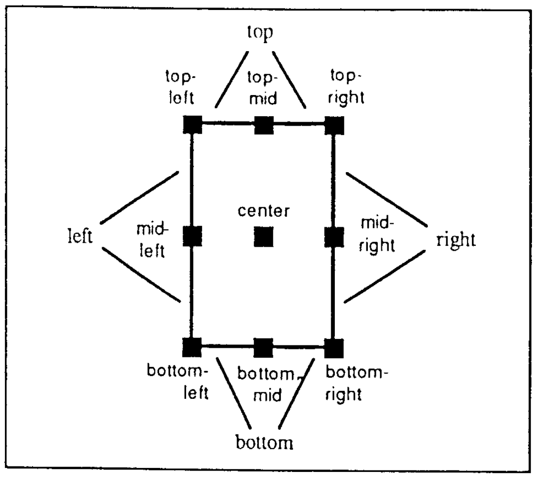
e. Repeat c, d from N

f. output

Figure 3. Task: given a set of points, find their convex hull.



a. parts of a line



a. parts of a box

Figure 4. Parts of objects.

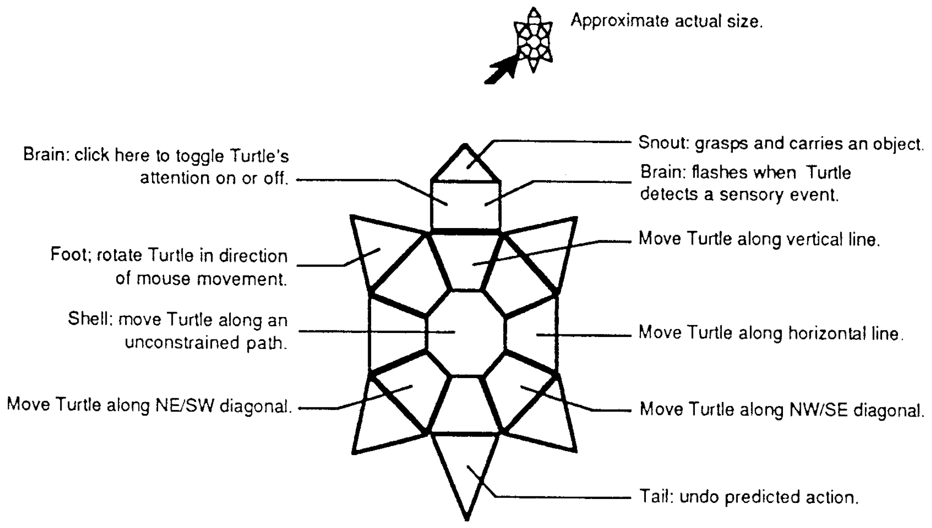
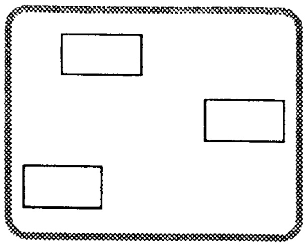
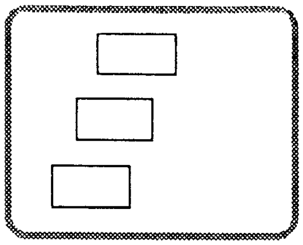

Approximate actual size.
Brain: click here to toggle Turtle's attention on or off.
Snout: grasps and carries an object.
Brain: flashes when Turtle detects a sensory event.
Foot; rotate Turtle in direction of mouse movement.
Move Turtle along vertical line.
Shell: move Turtle along an unconstrained path.
Move Turtle along horizontal line.
Move Turtle along NE/SW diagonal.
Move Turtle along NW/SE diagonal.
Tail: undo predicted action.

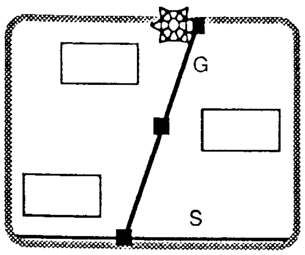
Figure 5. Anatomy of Meta-Mouse.



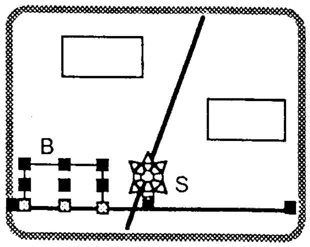
a. before



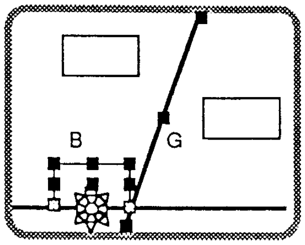
b. after



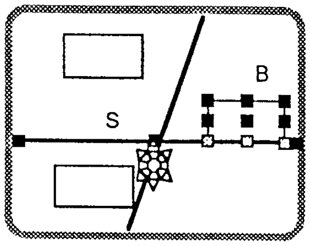
c. create guideline G and sweep-line S



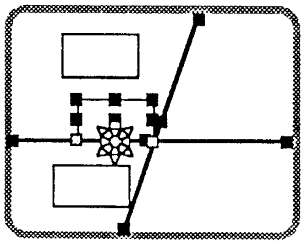
d. move S upward until contact with a box B



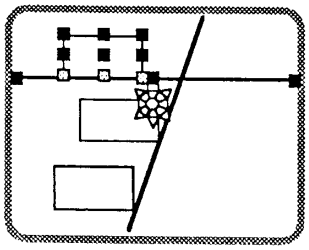
e. move B right until contact with G



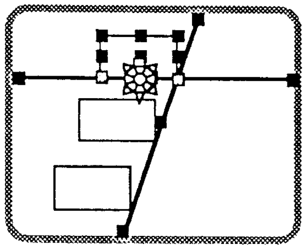
f. action as in d; heading generalized.



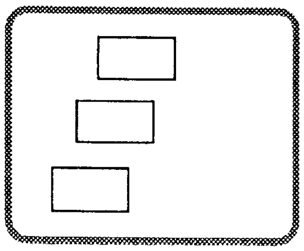
g. Turtle predicts action as in frame e.



h. Turtle predicts action as in frame d.

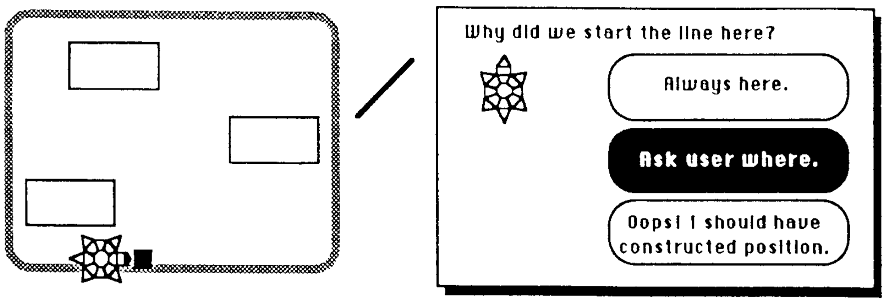


i. Turtle predicts action as in frame e.

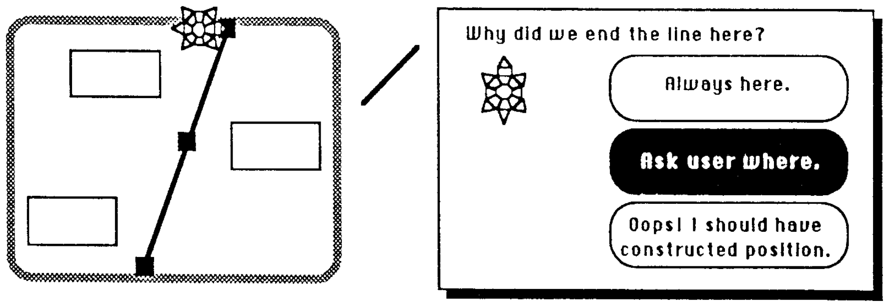


j. remove S and G, done

Figure 6. Task: align boxes ("box-to-line"), complete with action/teaching trace


Why did we start the line here?
Always here.
Ask user where.
Oops! I should have constructed position.

a. query for explanation of line origin


Why did we end the line here?
Always here.
Ask user where.
Oops! I should have constructed position.

b. query for explanation of line termination

Figure 7. Asking the teacher to explain seemingly arbitrary decisions.