

16. Sketching a Multidisciplinary Microworld: A Collaborative Exploration in Boxer

Jeremy Roschelle¹ and John Mason²

¹University of Massachusetts, Dartmouth, MA USA

²Open University, UK and Sunrise Research Laboratory, Melbourne, Australia

Abstract. In this chapter, we explore the possibility of designing microworlds that support exploration of overlapping perspectives of two (and perhaps more) disciplines. Mathematical and computational points of view are presented together in a Boxer microworld called “ClockGame.” We draw attention to the sketch-like properties of the microworld, and the possibilities this creates for student learning. A second point emphasizes our construction process; we give accounts of how sketching a learning environment together provides opportunities for collaborators to learn from each other.

16.1 Introduction

Interactive media allow the possibility of designing microworlds that support exploration of overlapping perspectives of two (and perhaps more) disciplines. The benefit of collaboration among disciplines is perhaps obvious: Most interesting problems require collaboration among people who can simultaneously bring two or more disciplines to bear. However, obstacles to collaboration can be hard to overcome.

We draw attention to one particular requirement of multidisciplinary work: Collaboration requires a common medium (Roschelle, 1992, 1994). Collaborators need not share the same theories, nor the same methods, but they do need a medium in which they can act jointly. Sometimes this medium can be as simple as using words to have a conversation, or a few hastily sketched figures. However, as advocates of exploratory learning well know, it is often not enough to be able to tell your ideas to another person.

In many design practices, collaborators use sketches to explore ideas with each other (Schön, 1987). In this paper we explore the sketch metaphor in relation to the Boxer computational medium. Boxer is a computer system and language intended to provide a medium for developing educational microworlds (diSessa, this volume). Boxer, we argue, provides a powerful environment for multidisciplinary collaboration because it enables collaborators to sketch prototypes with each other.

Boxer thus provides an authoring medium in which collaborators can more easily inhabit the same design space (Nevile, this volume). The experience of *collaboratively constructing* a prototype in a shared design space can be a powerful basis for further intellectual teamwork (see Galegher, Kraut and Egido, 1990).

In providing a microworld with both mathematical and computer science components, our wish is to provide just enough detail to awaken the users' desires to explore further in either domain, and to invoke their own powers of generalization and construal. The idea is to avoid as far as possible the *transposition didactique*, in which expert awareness is transformed into elaborate instruction (Chevellard, 1985). In Mason (this volume), this transposition is related to a distinction between a painting and a sketch. The painting (traditional instructional text) has to be worked at, dissected, and reconstructed in order to see past the completed exposition to the important component details and principles, whereas a sketch invokes viewers' gestalt powers of closure and completion to construct their own artifact based on the sketch.

In this paper, we examine the Boxer medium as an environment for providing both students and colleagues with sketches that invoke construal of important ideas. We report on an occasion in which Boxer became a powerful medium for collaboration among the two authors and their colleagues. John, a mathematician, came to a Boxer workshop with a mathematics problem, which he initially presented as a verbal sketch. Jeremy, a computer scientist, was using Boxer to share computer science concepts with a group of colleagues. He saw John's idea as a valuable opportunity to share important programming concepts. Mathematical and computational points of view came together in a Boxer microworld called "ClockGame."

From John's verbal description, a ClockGame prototype was constructed in the space of one morning. This working model/sketch provided valuable opportunities for members of the workshop to learn from each other. By playing with ClockGame, Jeremy found an opportunity to explore the ideas of group theory in an engaging microworld. John likewise found an opportunity to explore the computational ideas available in Boxer with respect to an area of personal interest. Our colleagues found in ClockGame a provocative example of how Boxer can provide a variety of learning experiences, with either mathematics or computer science coming to the foreground depending on what a learner chooses to explore.

This chapter strives to make two points with the sketch metaphor. One point focuses on the product; we use ClockGame to show how Boxer designers can build microworlds that support multidisciplinary exploratory learning. Here we draw attention to the sketch-like properties of the microworld, and the possibilities this creates for student learning. A second point emphasizes the process; we give accounts of how sketching a learning environment together provides opportunities for collaborators to learn from each other. By sketching working prototypes of ideas in Boxer, collaborators can educate each other's awareness of their home disci-

pline, provoke construal of disciplinary ideas, and enable multidisciplinary design to draw upon an enriched base of shared understanding.

16.2 Boxer ClockGame

In its pure form, the ClockGame is a mathematical puzzle: Arrange a set of clocks in a matrix with the same number of rows and columns and then with a limited set of moves available, try to make all the clocks read the same. For background to the puzzle, elaboration of some mathematical aspects, and a different implementation, see Mason (this volume). The clocks have one hand which can occupy a small number of positions. For beginners a 3x3 grid of clocks with hands that point either up, down, left, or right works well. Two operations are available on the clocks: flip a row and flip a column. Here “flip” means rotating the hands of all the clocks in the row or column to the next clockwise position. The goal of the puzzle is to use flip-row and flip-column operations to move from a random arrangement of clock hands to a pattern in which every clock hand points in the same direction.

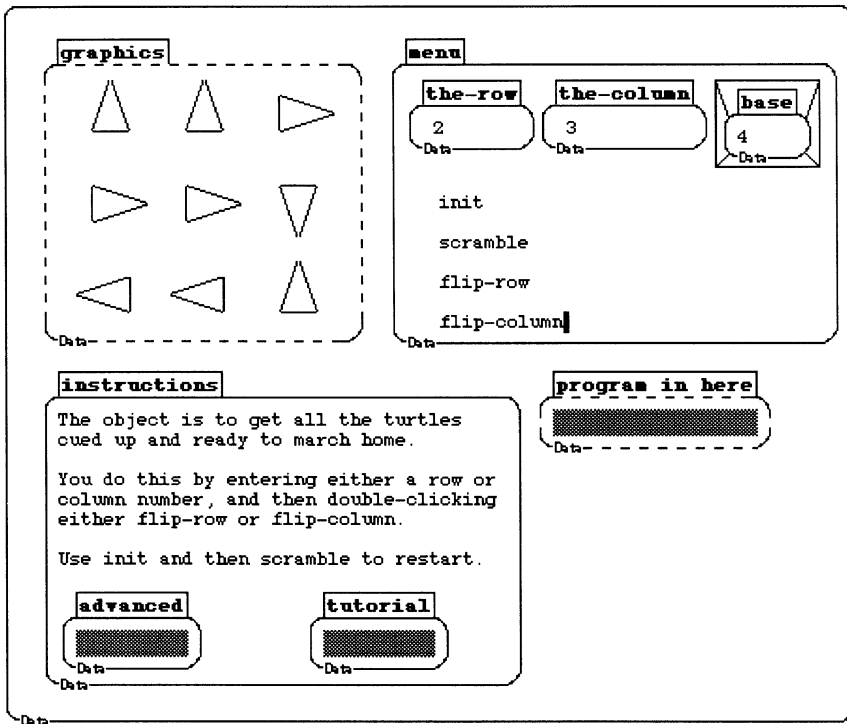


Figure 1. Boxer ClockGame.

Like other microworlds (see Edwards, this volume), ClockGame presents a set of computational objects that reflect the structure of a knowledge domain. Rather than presenting a complete curriculum, the microworld challenges students to make sense of the functioning of a system; in grappling with the behaviors they experience in ClockGame, students will be working on concepts that lead towards group theory (see Mason, this volume, for details).

The Boxer ClockGame uses a matrix of turtles to depict the clocks. A menu, which appears to the right of the game board, provides several commands. The `initialize` command makes all the turtles face upwards. The `scramble` command carries out a random number of random operations to set the clock hands to different directions. The `flip-row` and `flip-column` commands use the number in the row or column data boxes, and flips the corresponding row or column.

To play the game, the user first clicks on `initialize`, and then `scramble`, producing an initial puzzle state. The player then chooses row or column flip operations by typing in the desired row or column number and clicking to execute the command.

In addition to playing the game, our design supports explorations of the mathematical and computational structure of the ClockGame. Like Eisenberg (this volume), we give learners procedures and variables which they can inspect and modify. We supply procedures to change the number of clock positions and the number of clocks. A variable called `base` allows the user to set the number of possible clock positions. As we will later describe, the initial 3x3 grid of clocks can be readily extended to 4x4, 5x5, or larger grid. To further provide scaffolding for computational exploration, we have included an expanded tutorial lesson (described later).

16.3 The Sketch Metaphor and Boxer

Boxer is a reconstructible computational medium (diSessa and Abelson, 1986)—a computer system intended to allow authors to explore ideas in a computational form. A rich medium, however, should not be limited to exploration. Hence, Boxer is also a valuable medium for expounding and explaining, exploring and examining, expressing and exercising. Boxer provides a rich authoring medium with multiple modes of expression readily available to different authors.

Boxer incorporates two important design principles, spatial metaphor and naive realism (diSessa and Abelson, 1986). Spatial metaphor refers to Boxer's extensive use of spatial relations to organize computation. In particular, Boxer depicts data and programs as *boxes*. Boxes can be contained within other boxes, and can be shrunk to a small icon, thus hiding their contents, or expanded to full screen size to allow exploration of their contents. The box is a powerful structuring device for rapidly designing a complicated interface; non-programmers can sketch a set of

nested Boxer screens that contain diverse interactive elements such as menus, graphics, instructions, tutorials, etc.

Naive realism means that Boxer enables the user to treat the screen as if it depicted the state of the computational system. Thus, Boxer tries to alleviate the need for users to imagine objects and processes that operate “behind the screen.” Instead Boxer provides a visual model that directly supports a user’s mental model of the system. Simultaneously, naive realism reduces the distance between design sketches and a working model. For example, a designer can sketch out a set of parameters and menus in a Boxer microworld by creating a box for each one. A programmer can quickly turn these boxes into an actual working implementation.

Finally, a mundane but crucial factor in multidisciplinary teamwork is time. Boxer’s design enables collaborators to assemble a working prototype quickly, which then can serve as an object for reflection and elaboration. Boxer provides a diverse set of ready-to-use tools for graphing, data bases, word processing, microworld construction, etc. In contrast, in many authoring media, each of these tools would have to be separately developed, and this might take a long time. In our case, the ClockGame prototype was built in two days that John and Jeremy spent together at Sunrise Research Laboratory in Melbourne, Australia. We were each able to take this prototype with us to continue exploring and elaborating afterwards. However, had we not been able to build a prototype in two days, our collaboration would never have gotten past the idea stage.

16.4 John: Exploring the Mathematics of Groups

Prior to our ClockGame collaboration, I had built an elementary version in the BASIC computer language. This required considerable effort. I proposed continuing to explore the game in Boxer because it provides an attractive logic puzzle, with enormous potential for mathematical exploration. Moreover, I personally wanted to explore some generalizations of the game which were too hard to do theoretically. I felt I needed a working “object to think with.”

When Jeremy took up the idea, he heard only part of the specification and interpreted the task to be to return the puzzle to its starting configuration when it was scrambled. After a period of doing this, as Jeremy reports shortly, it all seemed too easy. There is then the possibility of turning attention to the question of which configurations you can reach, and which are unreachable. This can lead to an important mathematical idea known as the orbit.

With the alternative task intended by John (see Mason, this volume), the puzzle provides an opportunity to get stuck, even frustrated, by not being able to achieve what seems simple. Then the student can stand back and recognize that there are a series of operations available on the clocks, and that there is an operation on those operations, namely composition. (Composition is combining several moves to make

a single one.) Using the operations rather than the clocks, the remarkably small number of possible states of the clocks becomes apparent. Awareness of operations on operations was suggested by Gattegno (1987) as the essence of algebra. In this case it affords access to the algebraic structure known as a group. Seeing flips as both operations on rows and columns of clocks, and as objects which can themselves be combined constitutes an essential and important shift of attention (Mason and Davis, 1988; Mason, 1989) that is required in order to reason algebraically.

Furthermore, as Davis and Hersh (1981) mention, it is common in mathematical research to be unsure whether what you are trying to prove is actually true or false. In ClockGame, some tasks are impossible, and this becomes apparent during exploration. This provides an opportunity to draw attention to a subtle series of shifts from trying to succeed, to reckoning it to be impossible, to searching for a reason for it to be impossible (Mason, this volume).

16.5 Jeremy: What I Learned About Groups

Prior to working with John on ClockGame, I believe my only exposure to mathematical groups had occurred in the “New Math” that I experienced in secondary school. My understanding of groups was rusty and mostly forgotten, to say the least. So I initially experienced the ClockGame as a somewhat interesting puzzle that my creative mathematical colleague wanted to implement.

As I began to play with the ClockGame, I discovered some intriguing phenomena. First, I found that I could always return the clocks to their original orientation in surprisingly few moves. Using Boxer’s `repeat` command, I tried executing the `scramble` procedure a thousand times. Contrary to my expectations, the game never got “more scrambled”—it always seemed to take a half dozen or so moves to return the clocks to their original position. Second, I replaced the `scramble` command with a command that set individual clocks to random orientations (recall that the `scramble` command uses the flip operators, and cannot change clocks individually). To my consternation, I discovered that it was now usually *impossible* to return the clocks to all face upwards, no matter how many flips I executed. Lastly, I explored playing the ClockGame with a 4x4 and a 5x5 matrix. I assumed that this would make the game much harder. Wrong again. These larger grids did not appear to get much more scrambled than the smaller grids.

At last John came to my rescue, reminding me of some of the basics of groups—a set of operators that produce a closed set of values, with properties such as inverses, commutativity, etc. As I continued to explore ClockGame, I found that I could now make some sense of the puzzles. It is always fairly easy to complete the puzzle because there is actually a fairly small set of possible board states, and the maximum number of operations to go from the initial state to the farthest possible board state is not very large. This is because repeating a particular operator enough

times always produces an identity transformation. For example, in the case where the clocks have two positions, repeating any operator an even number of times is the same as not executing it at all, and repeating it an odd number of times is the same as executing it once. It is impossible to solve the puzzle every time in the random case because individually randomizing the clock may produce an element outside the group orbit. The larger board sizes do not increase difficulty much because they introduce only a few new operators. Thus I found ClockGame to provide a practical context for exploratory learning in which I could discover the power of some ideas from group theory.

16.6 Jeremy: Exploring Computation with Data Representations

One of Boxer's virtues is that it removes barriers between the programmer and the user of a microworld (see Eisenberg, this volume). On the one hand, the user can shrink all the boxes containing the programming code for a microworld, and just focus on experiencing the microworld itself. On the other hand, a user who wishes to explore computer science concepts can proceed to expand the boxes containing program code and explore their workings. Figure 2 shows the ClockGame with the "instructions" box closed and the "program in here" box expanded.

The ClockGame provides an excellent opportunity for beginning Boxer programmers to learn about a very important computing concept: the power of choosing a good data representation. As is evident in Figure 2, ClockGame contains just a few lines of code. This simplicity, however, is possible only because of a clever exploitation of Boxer's data representation capabilities. Thus, exploring the code may not be revealing to the novice programmer.

To support exploration of this computing concept, we created a tutorial box within the ClockGame microworld. Expanding the tutorial box (Figure 3) leads to a structured presentation of the computing concepts at work in the game. The presentation builds up from a simpler microworld example to the actual ClockGame. This takes advantage of a unique feature of Boxer—the ability to embed an entire microworld within a box, and thus encapsulate diverse and complex supplementary experiences within the main microworld.

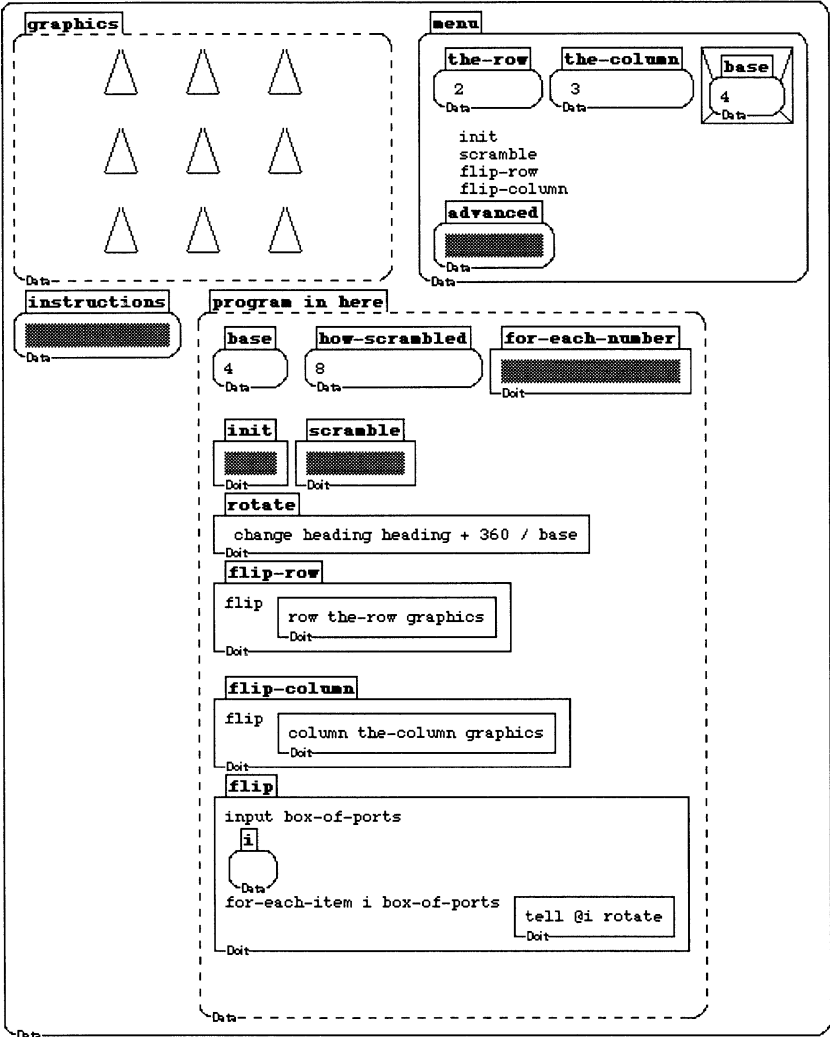


Figure 2. Exploring the program code.

tutorial

Browsing and thinking about this microworld should open up a new way of thinking about Boxer programming for you -- in particular, how you think of graphics boxes.

But first, it is important that you think for yourself about how you would write the microworld yourself. Make a box for yourself on the line below inside how-I'd-do-it, and write a few lines about what is obvious to you and what would be hard.

how i'd do it

Elizabeth, Tony and Anne's effort

What we could do :
 set up a turtle box
 make and position 8 additional sprites that look like turtles
 base-setting procedure relating angle turned in scramble command to base no.
 scramble, containing random turns of this-base's angle for each sprite
 procedure for flipping row, using this-base's angle and row no. as inputs
 ditto for columns
 What we couldn't do:
 we'd have to write it thus far and see what else we needed

jon and angela **joy and paul**

simpler world <----- After you've thought about it, go in here

fancier world <----- After you've read the simpler world, go in here.

Figure 3. The Tutorial, including contributions of prior users.

Upon opening the tutorial, the learner is first asked to write down thoughts about how the ClockGame works. Most Boxer users immediately recognize that the clocks are turtles. However, the ClockGame radically breaks from the most common model of programming with multiple turtles. In the common model, users give each turtle a name, and direct commands to individual turtles with the `tell` primitive. For example, to flip the second column a novice programmer might write:

```
tell turtle-1-2 rotate
tell turtle-2-2 rotate
tell turtle-3-2 rotate
```

Using this method, each row or column flip requires a separate set of three commands that refer to the correct subset of turtles. By having learners reflect on their expectations before starting, we set the stage for provoking an insight—that a program can refer to turtles not by their name, but by their location within a data

structure. By using data representations to structure the program, rather than arbitrary names, a Boxer programmer can take advantage of Boxer's powerful primitives for operating over data representations.

The presentation is structured to work gradually up to this important lesson as the learner explores a series of boxes. The first box (Figure 4) contains a simpler version of the ClockGame microworld. In this version, the game board is represented as a matrix of numbers, rather than graphical clock hands. The learner can experiment with this microworld using the menu of commands. As in the graphical ClockGame, a main command called flip operates over both rows and columns.

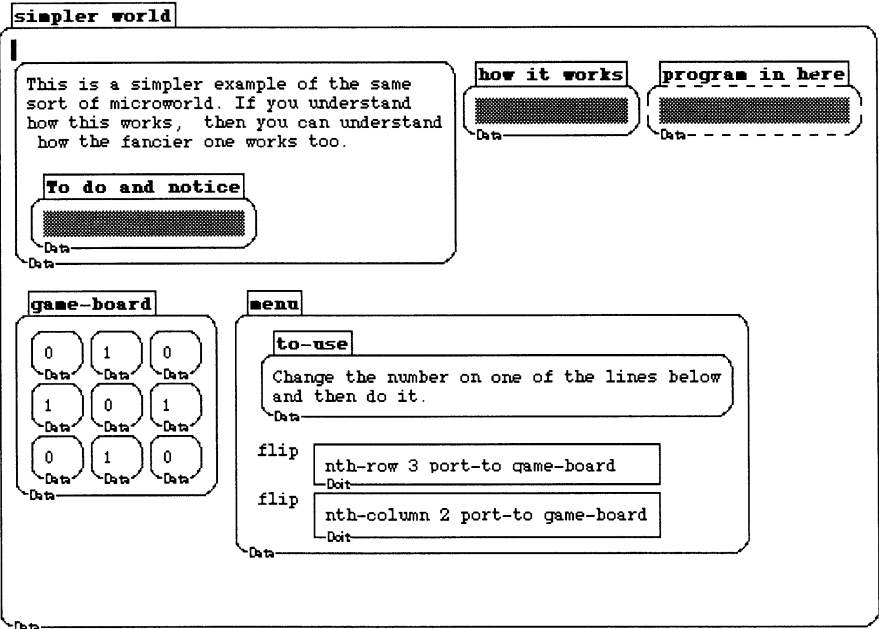


Figure 4. Exploring a simpler microworld.

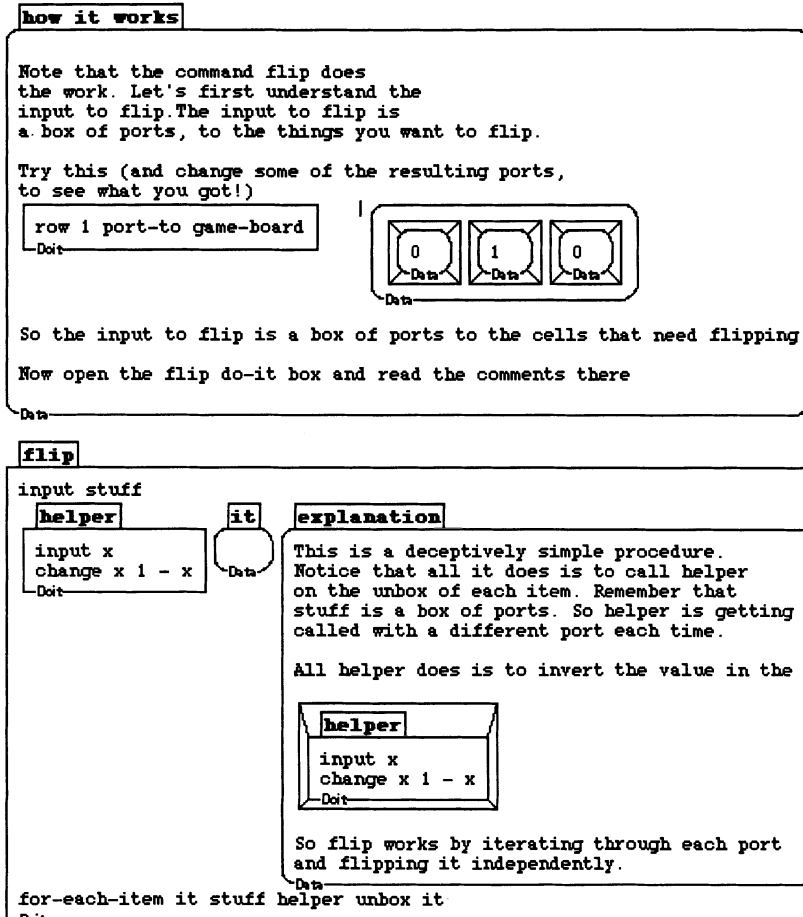


Figure 5. How the Simpler ClockGame works.

The tutorial contains much less text than a traditional tutorial, but considerably more opportunity to experiment with concepts interactively. A conventional tutorial is more like a painting, chock full of all the details that constitute the author's understanding, and so *must* be provided to the student. Mary Boole (Tahta, 1972) called the tutor's urge to explain "teacher lust," and there is much in the metaphor. In contrast, the ClockGame tutorial is more like a sketch: It hints at the major ideas, and encourages interactive exploration of several key programming constructs, but it does not try to explain ideas fully. In particular the tutorial may not explain ClockGame sufficiently to the reader of this chapter, especially without the benefit of an interactive medium.

The tutorial organizes the students exploration into two parts. First, the user opens the box called “how it works.” This box (Figure 5) examines the input to the `flip` command. `row` is a built-in Boxer procedure that returns a single row from a box. In this experiment, we get the first row of the game board. The value returned by the `row` command appears to the right of it, after the horizontal line in the top box in Figure 5.

Second, the user opens the “flip” box. This shows another style of providing tutorial help in Boxer. Rather than surrounding the procedure with text, help is embedded within the procedure, in the form of a data box. The learner could read this text, and experiment with how the procedure works by executing it in a step-by-step fashion.


The learner would next close the `Simpler ClockGame` and open the next box in the tutorial, which explains how this programming strategy is used with graphics (Figure 6). This portion of the tutorial is structured as a set of questions and answers. Closed boxes are used to hide the answers until the learner chooses to open them. The key question is “What does `row` of a graphics box return?” since the `flip` procedure will simply apply a rotation to each element of its input. The answer, shown to the right in Figure 6, explains that the graphics box has been structured so that the data box for each sprite (turtle) is in the appropriate location in a matrix. Thus the `row` command returns a box containing just those sprites in a particular row. Once the appropriate subset of sprites is selectively retrieved, then the `flip` command can rotate them by iterating through each one.

This programming method has many advantages over the more obvious method in which each turtle is accessed by giving it a unique name. First, it reduces the amount of code because Boxer’s generic `row` and `column` selectors retrieve the appropriate subset of objects, rather than special procedures written by the programmer. Second, the resulting `ClockGame` is easily generalized to larger matrices. Indeed, as long as the sprites are placed in the graphics box in a matrix formation, any size matrix will work without any modification to the `ClockGame` code. This is not possible in the novice version in which each sprite is referred to by name. Third, the method generalizes nicely from numbers to graphics, and from there, to any sort of object that could be represented by a box.

The tutorial provides an opportunity for interested learners to make a structured exploration of how this method of data representation is deployed in the `ClockGame`. The Boxer principle of spatial metaphor, in this example, organizes the exploration into a series of boxes that progressively elaborates the central idea. The Boxer principle of naive realism is used to make the workings of the system concrete, executable, and manipulable. For example, the learner can open each data and `doit` box (i.e., variable and procedure) to see its contents, and can watch variables change as the program executes.

fancier world

The fancier microworld works by exactly the same principle as the simpler one. Note that flip-row and flip-column just call flip with a subset of what is in the graphics box.

 <---open and read these ports to the procedures.

So we have 2 questions:

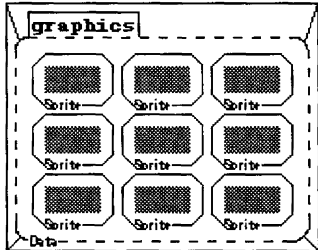
What does nth-row of a graphics box return?

answer

Well it depends, doesn't it?

It depends on what is in the graphics box. Take a look inside this port.


port-to graphics



What I've done is to arrange the sprites in rows and columns, so that their spatial layout on the graphics side and their spatial layout on the data side correspond. Then, rather than naming the sprites, we can use nth-row, nth-column, or rc to get particular ones.


nth-row 2 port-to graphics
 nth-column 1 port-to graphics

Using this we can give flip a box full of ports to sprites.




What does flip do with that row?

answer2



One more question: how did I get the spatial locations to correspond?

answer3






Figure 6. How the ClockGame works with graphics.

16.7 John: What I Learned About Computing in Boxer

I was struck by the cleverness of being able to use a structural (spatial) reference rather than having to label everything. I liked the idea of acting directly via ports rather than on names. It seemed a little mysterious at first, but once I had worked through Jeremy's example I was able to modify it to build my own clocks and

develop them in the direction that interested me. I still have a sneaking feeling that ports are in some sense “expensive,” but, as I use them more and more, I have begun to see how effective they can be. Often when I am not succeeding with building a procedure, it is because I am not using ports! I had actually learned my lesson well.

16.8 Discussion: Supporting Multidisciplinary Exploration

Most microworld environments support exploration of only one discipline. Using the ClockGame example we have shown how Boxer enables the design of microworlds that can enable learners to explore two disciplines, mathematics and computer science. Other disciplinary combinations are possible. For example, David Williams has examined students’ use of Boxer to explore a mathematical model of the spread of disease (Williams and Roschelle, 1993). These students learned simultaneously about mathematical tools such as graphs and tables of numbers, and biological ideas concerning populations and infectious disease processes. Another combination is science and writing. A substantial amount of prior Boxer work has been concerned with the exploration of kinematics concepts (e.g., diSessa, this volume). One group of students simultaneously explored motion and writing: They elaborately narrated an adventure game in which the hero narrowly escapes a series of traps by planning the correct velocity and acceleration. Here the spatial metaphor was used to contain parts of the story in closed boxes that can be progressively opened. Each part of the story contained both text and a running microworld.

Creating multidisciplinary exploratory learning opportunities requires collaboration among microworld authors with different academic backgrounds. But it also requires an *authoring medium* that enables a team of authors to express and experience ideas in diverse forms. We believe that Boxer’s design provides an exceptionally powerful authoring medium for multidisciplinary exploratory learning. Below we highlight some of Boxer’s features that enabled our work.

Rapid Prototyping

Boxer makes it easy for a design team to move quickly from a verbal description of their target microworld to a sketch of implementation, and from there to a working model. Boxer facilitates prototype construction by providing ready-to-use capabilities for menus, graphics, text, and for structuring tutorials. Moreover, because the medium is easily learned, programmers and non-programmers can more easily author together. Here we give a small additional example. As mentioned earlier, learners can explore clocks with more than two positions by changing the base variable. It took *no additional programming* to make this opportunity for generalizing available to John, the mathematical subject matter expert. Because of naive realism, Boxer simply shows the variable itself as a box on the screen. By typing in a new value, John could immediately change the actual setting for the number of clock positions as used by the program code. He used this capability to

show Jeremy how students might explore generalizations of the initial clock game. Thus, in Boxer, a mathematician can make small modifications to the work of the programmer, and the programmer can explore mathematical ideas in a set of Boxer experiences prepared by a mathematician.

Spatial metaphor

Box structure provides a means to control the complexity of the display, and thus to keep the details of multiple paths of explorations from overwhelming the learner. In our case, the computer science tutorial could be contained within a closed set of boxes, thus hiding this detail from a learner who prefers to study the mathematics. In our work, spatial metaphor allowed us to organize the exploration of several ways in which the ClockGame might grow by grouping those explorations into nested boxes. Collaboration is made easier when a medium provides a way to add details to the prototype while preserving the simplicity of the basic idea.

Naive realism

The Boxer principle of naive realism holds that abstract systems are easier to understand when rendered as concrete objects that can be inspected and manipulated directly. Boxer implements naive realism by removing the boundary between the programmer and the user; users can always open their boxes to see how a microworld works, or close them to focus on the subject matter of the microworld. This readily enables computer science lessons to be combined with subject matter lessons. The tutorial exploits naive realism by taking learners for a guided tour “behind the screen” of the ClockGame microworld. We also think naive realism is important for collaborating authors because it reduces the barrier between experts in programming and experts in subject matter.

In closing, we highlight the idea that *jointly constructing* a microworld can be a satisfying multidisciplinary experience. The opportunity to build something new helps people learn (Harel and Papert, 1991). In this case, sketching the ClockGame created a rich context for collaboration between a mathematician and a computer scientist. By participating in the gradual refinement of a microworld, each author observed how his collaborator approached and grappled with a problem, not just what his finished solution looked like. This provided opportunities for each author to gain some experience with the working methods of the other discipline. Producing a joint product results in collaborative experience that is satisfyingly concrete.

In future work, we would like to see how opportunities for collaborative construction of a multidisciplinary object could be extended to students, who might build as well as explore multidisciplinary microworlds. The key element is to expose students to opportunities and possibilities rather than trying to train them in particular behavior. This might provide opportunities for students with different abilities and inclinations to participate more effectively in collaborative learning.

Acknowledgments

We thank Liddy Nevile at the Sunrise Research Laboratory, Royal Institute of Technology, Melbourne, Australia for both supporting and contributing to our collaborative work.

References

- Davis, P.J. and Hersh, R. (1981) *The Mathematical Experience*, Harvester
- diSessa, A.A. and Abelson, H. (1986) Boxer: A reconstructible computational medium, *Communications of the ACM*, 29, 859-868
- Chevellard, Y. (1985) *La Transposition Didactique*, Grenoble: Le Pensée Sauvage
- Galegher, J., Kraut, R.E., Egido, C. (eds.) (1990) *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, Hillsdale, NJ: Lawrence Erlbaum
- Gattegno, Caleb (1987) *The Science of Education, part I: Theoretical Considerations*, New York: Educational Solutions
- Harel, I. and Papert, S. (eds.) (1991) *Constructionism: Research Reports and Essays*, Norwood, NJ: Ablex
- Mason J. and Davis J. (1988) Cognitive and metacognitive shifts, in Borbás (ed.), *PMEXII*, 2, 487-494
- Mason, J. (1989) Mathematical abstraction seen as a delicate shift of attention, *FLM*, 9/2, 2-8
- Roschelle, J. (1994) Collaborative inquiry: Reflections on Dewey and learning technology, *The Computing Teacher*, May, 6-9
- Roschelle, J. (1992) Mediated collaborative inquiry: Towards a practical philosophy of learning technology, in M. Ryan (ed.), *Classrooms: Where Ideas Meet Reality*, Brisbane, Australia: Queensland Society for Information Technology in Education
- Schön, D. A. (1987) *Educating the Reflective Practitioner*, San Francisco: Jossey-Bass
- Tahta, D. (ed.) (1972) *A Boolean Anthology: Selected Writings of Mary Boole*, Derby: ATM
- Williams, D. and Roschelle, J. (1993) Mathematical problem solving in a Boxer computational environment, in Jaworski, B. (ed.) *A Bridge between Teaching and Learning*, Technology in Mathematics Teaching Conference Proceedings, Birmingham, UK, 491-498