Exploring Human Visualization of Computer Algorithms

Sarah Douglas[1]
Donna McKeown[2]
Christopher Hundhausen[1]
[1]Computer & Information Science Dept.
[2]Psychology Dept.
University of Oregon
Eugene, OR 97403
(503) 346-3974
douglas@cs.uoregon.edu

ABSTRACT
Many educators have used Algorithm Visualization (AV) to teach students of computer science about how computer algorithms work. Our study sheds light on two important questions: (a) How do people conceptualize algorithm animations in the first place; and (b) To what extent do such visualizations accord with AV software. In the first half of this study, pairs of graduate students in computer science were asked to construct animations for a simple sort (bubble sort) using ordinary art materials. In the second half, they implemented a bubble sort visualization using an interactive AV program called LENS [1], which allows one to construct and view an animation of any C program. The way in which pairs visualized the same sort differed tremendously from each other and did not accord completely with the animation language provided by LENS. This paper analyzes those differences by a detailed examination of the semantics of the human visualizations, the algorithm code, and the LENS AV language.

KEYWORDS: visualization, algorithm visualization, mental models, empirical studies, individual differences

INTRODUCTION
Algorithm visualization (AV) can be defined as the process of viewing an algorithm through a series of pictures, or through a movie. In AV the algorithm's dynamic procedural behavior is represented as state changes to graphic entities. This illumination of the logic that underlies algorithm behavior has been used for communicating concepts about the algorithm and for testing correct implementation. Pioneered by Brown University's Electronic Classroom project [2], and formalized by Marc Brown's seminal dissertation *Algorithm Animation* [3], AV has gained an enthusiastic following among undergraduate computer science educators who have come to see it as an effective and innovative method for teaching algorithms. In a culture in which the proverb "A picture is worth 1,000 words" is held so dearly, it should be no surprise that a learning technique designed to exploit the visual sense has been embraced so widely; indeed, thus far, its powerful intuitive basis has been sufficient to justify its use.

Despite this enthusiasm, AV's effectiveness in learning algorithms has not been demonstrated empirically. For example, in one of the few experimental studies to date Stasko, Badre, and Lewis [4] published disappointing results. In the Stasko et al. study computer science graduate students taking an advanced algorithms course were presented with information about a pairing heap algorithm in two different media: text-only versus text-and-animation. The text-and-animation condition was presented on the XTango animation system. XTango is derived from Tango [5] and provides minimal interactivity by allowing speed control, but not rewind or replay capability. Learning was measured by a test of 24 questions. These questions reflected three different types of knowledge and thinking: declarative, analytical, and procedural. It was hypothesized that the text-and-animation condition would particularly benefit the acquisition of procedural knowledge. Unfortunately, the results of the study as measured by test scores revealed no significant advantage for AV. In fact the test results for both conditions indicated poorer performance than expected, with the animation group showing no advantage in the test of procedural knowledge.

The authors cite two reasons for these results. One was primarily methodological: the presentation materials in either condition failed to provide students with enough information to be able to reconstruct enough of the procedural elements of the algorithm to do well on the test. More importantly the authors feel that understanding the algorithm may be prerequisite to understanding the passive viewing of an animation of the algorithm—clearly, a situation that learners do not enjoy. Most systems implementing AV, including the XTango system cited above, present an animation predefined by a human teacher who already understands the algorithm on which it is based.

In addition to understanding the underlying concepts of the dynamic process, students also must understand the relationship between those concepts and the graphical

representation. For example, in an AI tutoring system for teaching about the cardiovascular system [6], we generated a dynamic graphical network of cause and effect relations between variables such as pressure, flow and resistance to create visual explanations of simulation behavior. Students had difficulty understanding this predefined graphical representation even though they demonstrated knowledge of the process being illustrated. That is, a graphical representation as a "language" per se must be understood before it can aid in explanation.

Stasko et al. [4] describe this as a mapping from the abstract computational algorithm domain to the animated computer graphics domain. For example, in the movie *Sorting Out Sorting* [7] the magnitude of the elements being sorted are represented by different length and color of rods. Our prior research suggests that both the graphical language elements and the mapping may not be self-evident. The trick, then, for an AV system is to get the graphical representation right.

To salvage the use of AV for novice learners the Stasko et al. study proposes that future algorithm animation systems research must focus on ways to support students constructing their own animations with the purpose of promoting an 'active learning' process as opposed to the 'passive learning' of watching the animation produced by someone else.

In cognitive terms how would algorithm learning be enhanced by the interactive construction of an animation by novice learners? We can suggest two ways: abstracting essential features of the algorithm focuses the learner on high level concepts about the algorithm, and the construction of a visualization enhances remembering those features. Indeed, the psychologist Paivio argues that we have more than one kind of code for representing constructs [8]. Specifically he proposes both a verbal and visual code. The more codes we can relate the material to, the better we can remember it.

This suggests the following steps for the use of an interactive AV system:

(1) Conceptualize the algorithm through text and pseudocode;

(2) Conceptualize a visualization for the algorithm;

(3) Map that conceptualization to the AV graphical language; and

(4) Manipulate the AV system's interface to program the animation.

Given this, successful usability of future interactive AV systems depends critically on understanding more about how people visualize algorithms and whether those visualizations can be mapped onto AV language elements. Thus, as a plausible next step in research, we decided to conduct a qualitative empirical study of expert users and their naive visualizations of a simple sorting algorithm. We chose expert users since there was a certainty that they already understood the algorithm. Since we are also interested in gaining a better understanding of the usability of AV software which allows users to construct their own algorithm animations, we continued our study with participants learning and using an experimental prototype of a highly interactive version of XTango called LENS [1] to produce animations.

Our study sheds light on two important questions: (a) How do people conceptualize algorithm animations in the first place; and (b) To what extent do such visualizations accord with AV software. The focus of this paper will be primarily on conceptualizations; however, we will also consider how these impact the design of any interactive AV system using LENS as a prototypical example.

EMPIRICAL STUDIES

Our approach was to present participants with bubble sort, an algorithm with which they were already familiar. We then asked them to construct an animated visualization. Some of the participants went on to use LENS to animate the same sort and one other.

Study 1: Constructing Paper Animations

Participants

We videotaped three same-sex pairs of participants (two consisting of women, one consisting of men) during 45 to 70 minute sessions. All participants were graduate students in computer science at the University of Oregon, and all had prior experience with the C programming language and the X Windows environment. Two pairs had no prior experience with AV, and one pair's experience was limited to a knowledge of the sorting animations pioneered by the film *Sorting Out Sorting* [7].

Procedure

We provided our pairs of participants with art supplies—a full spectrum of colored construction paper, scissors, and different colored pens—and presented them the following task: "If you were to explain the bubble sort algorithm to someone who had never seen it, how would you do it?

```
#include <stdio.h>
main()
{
  int n,1,j;
  int temp;
  int a[50];

  int count;

  printf("Input number of elts in array\n");
  scanf("%d",&n);

  printf("Enter the elements\n");
  for (count=0; count<n; ++count)
    scanf("%d",&a[count]);

  for (j=n-2; j>=0; --j)
   for (1=0; 1<=j; ++1)
    if (a[1] > a[1+1])
      { temp = a[1];
       a[1] = a[1+1];
       a[1+1] = temp;
      }
}
```

Figure 1. The C *bubble sort* algorithm that participants animated

Feel free to use any of these resources (construction paper, scissors, etc.) to assist you in your explanation." Prior to

the videotaping sessions, all participants reported that they had previously seen the bubble sort algorithm, and that they understood it. However, we provided participants with C code for their reference. (See Figure 1.) These "construction paper" sessions were videotaped and analyzed. If they decided to design a visualization for a sample data set, we recommended that they only step through a few of the iterations for simplicity's sake.

Method

We employed a qualitative research technique called *constructive interaction* [9,10], in which two persons are videotaped as they problem solve or perform tasks given by the researcher. By using two participants a situation of collaborative problem solving is created whereby each participant must inform the other in an explicit verbal and visual record about problems, causes, and solutions. In this study, we used constructive interaction to collect the data which we then reviewed using conversational analysis techniques developed by Douglas [11]. This provided an interpretation of participants' intentions, expectations and strategies.

Observations

As it turned out, in explaining bubble sort to a hypothetical novice, all participants made extensive and intriguing use of the art supplies with which we supplied them. In fact, all pairs of participants used the materials to create a homemade animation of bubble sort operating on a sample data set; they augmented their construction paper animations extensively with their own gestures and comments. Our most significant observation was that in no case could the homemade animations constructed by our participants be produced by LENS.

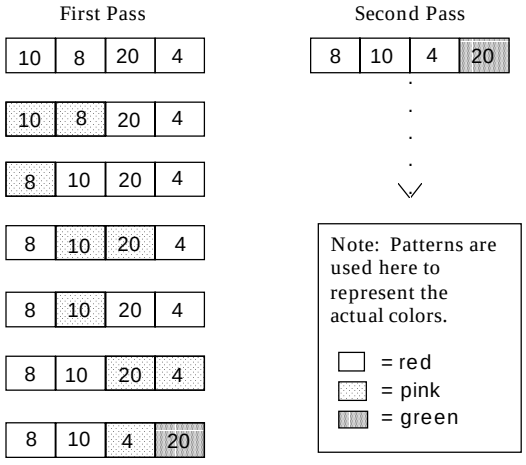

Figure 2. The model created by the Number Pair to demonstrate bubble sort.

One pair, we shall call them the Number Pair, used numbers (e.g. 5, 4, 3) to indicate the magnitude of each array element. This group used color to illustrate significant state changes. Figure 2 shows the color scheme which indicated whether an element was when two elements were being compared (turned pink), and to show the difference between sorted elements (turned green) and unsorted elements (red). Each pass of the array was then depicted as a (separate) horizontal row of (number-labeled) elements whose colors indicated what had happened during that pass. At the end of their demonstration, they had accumulated a history "array" of the sort, each of whose (horizontal) rows showed one pass of the bubble sort. A new column designated the start of a new pass.

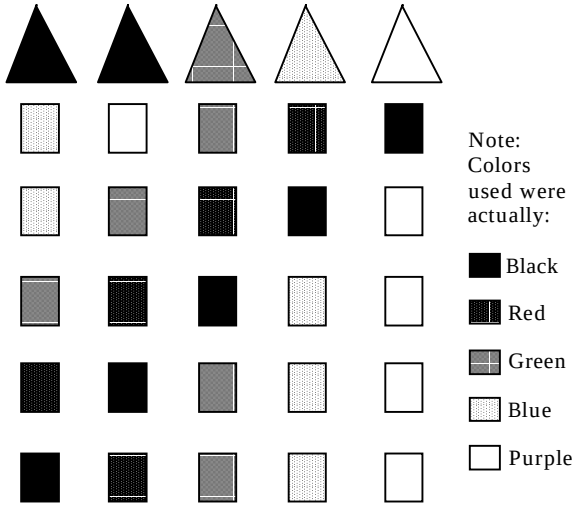

Figure 3. The model created by the Color Pair to demonstrate bubble sort.

In contrast, another pair, the Color Pair, used color to indicate magnitude; in particular, Figure 3 shows the legend they constructed (see triangles at the top of the figure) that defined a canonical (spectral) order for a five-element array. They demonstrated, by illustrating the array after each successive pass, how the bubble sort gradually placed an unsorted array into that ascending order.

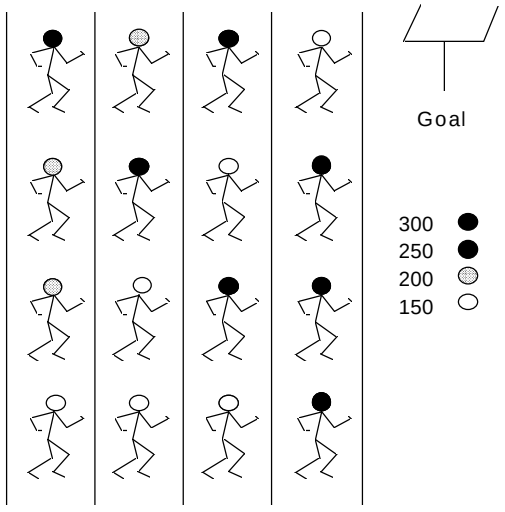

Figure 4. The model created by the Football Pair to demonstrate bubble sort.

Like the first pair, this pair had built, by the end of the sort, a history "array" in which each bubble sort pass occupied a row. Since they gave each array element its own color, the movement of array elements throughout the sorting process could be clearly seen.

Figure 4 shows how another pair, the Football Pair, also used color to denote magnitude. Their visualization depicted an American football game to describe bubble sort using the "weight" (or magnitude) of each player (or element). Players who weighed more "knocked over" the smaller players (they exchanged places in the array). The ball carrier and the player next in line symbolized what two elements were being compared at any given time. The first player was given "the ball" and started moving down the line of other players. If the next player weighed less, the ball carrier knocked the next player over and continued down the line. If the next player weighed more, then the ball was "fumbled" and the larger player now carried the ball with the previous ball carrier staying in the same place.

Discussion

Despite the variety of visualizations created by different pairs in this task, there were some common patterns. In all cases, subjects began to solve the task by first agreeing on how the sort worked. Two of the groups created sample data and stepped through exactly how the algorithm would react to specific elements. After this stage of defining the sort, pairs would begin to generate ideas on how best to describe the sort to a novice. They would mention the fact that the audience would not know certain ideas, such as pseudocode, or the "proper spectral order" of colors. In order to help others understand their animations, two of the groups constructed legends to show their audience how their colors mapped onto magnitude. Finally, all of the groups created a sample data set and used it to show exactly how the algorithm would react to specific elements.

Study 2: How do people's visualizations accord with AV Software?

Two of our three pairs of participants who constructed paper animations continued on to learn LENS and to implement the bubble sort and one other sort animation with it. The second sort animation task is not reported in this paper since its usability details are beyond the scope of this paper.

The major difference between LENS and traditional AV software (see, for example, [3, 5,12]) is that it provides the user with a set of graphic objects and transformation processes which allows the user to construct the visualization for an algorithm *interactively*, and *at runtime*. Written for XWindows, LENS allows users to load the source code of their C programs into an interactive environment, in which they occupy a scrollable source code window.

Using the *interesting events paradigm* established by Brown [3], users can interactively annotate their algorithms by (a) identifying points in their programs at which interesting events occur; (b) determining what graphical sequence would appropriately characterize those events; and (c) mapping the interesting events to graphics by manipulating the user interface. In targeting the activities of "high-level debugging, program testing, and refinement, not low-level debugging" (Mukherjea & Stasko, 1993, p. 3), the LENS system purports to provide direct-manipulation support for activity (c) noted above. The user interface to LENS offers a menu of animation primitives (flash, move, exchange, change color, change fill, create/delete image) which animate rectangles. These primitives map interesting events in the code to the underlying Tango AV language [5] on which LENS is based. The spatial extent (either height or width) of rectangles is automatically determined by the value of elements in C code arrays.

Procedure

Participants were given a three page description of procedural instructions for implementing a predefined animation for bubble sort defined by the authors. If implemented correctly this animation is illustrated in the final sorted state in Figure 5.

The C code for bubble sort given to them in Study One was already loaded into the system. Participants were shown how to define the "interesting events" in the algorithm code and then map them to graphical animation events.

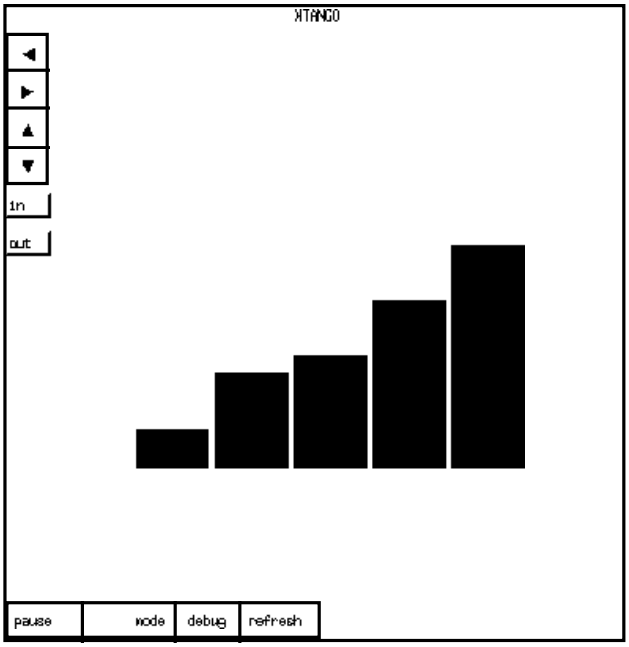

Figure 5. The LENS animation window which shows the final state of a dynamic bubble sort animation like the one participants defined in the study.

Users can choose to specify as many animation events as they wish using this process, and may even choose to specify multiple animation events on the same line. At any

point in the process of annotating an algorithm, users may view the animation currently defined on the algorithm causing an animation window to appear. Users may start the animation by clicking a button in that window. Thus, we see that LENS supports an iterative process of animation specification, in which users may interactively annotate the algorithm with animation primitives, and view, at any point, the animation to which those primitives give rise.

Observations

We are happy to report that, in both cases, our participants seemed to comprehend the bubble sort animation they defined; however, the participants did not immediately recognize that the animation was mapping array element magnitude to bar height (see Figure 5). In these cases, participants initially uttered remarks like "Huh?" and "What?," until it became clear that the bars were gradually falling into order, at which point they said something to the effect of "Oh, I see; height indicates value."

MAPPING VISUALIZATIONS

In order to analyze the visualizations of the human participants and the LENS system, we have adopted a mapping technique which relates conceptual entities, attributes and transformations in the algorithm pseudocode to the graphical entities, attributes and transformations of the visualizations. These graphical languages demonstrate both the similarities and differences in these multiple representations.

All of the pairs of participants created their own graphical language to describe the bubble sort in Study 1. Table 1 summarizes the mapping between entities and attributes defined in the algorithm and those created by each pair, and LENS. For example, the Number Pair used numbers in a row of contiguous squares to designate an array. LENS uses variable height or width rectangles which are drawn in a non-contiguous row. Some algorithm entities such as temporary storage and subscripts are not represented in the animation of either the human participants or LENS. Likewise, the legend of color codes, used by the Color Pair and the Football Pair to help the observer understand the correct order of sorted elements, is not available in LENS. Instead, LENS relies on the common-sense notion of increasing size. All human participants represented their animations as columns of rows of state changes. This history was not available in the LENS history. We do not know whether our participants did this because of the textbook use of this representation or because they wanted observers to have a visual history as external memory.

Table 2 summarizes the mapping between transformations defined in the algorithm and those created by each pair, and LENS. For example, the Number Pair used color to show when two elements were being compared (turned pink), and to show the difference between sorted elements (turned green) and unsorted elements (red). LENS used flashing to focus attention on the pair being compared and depended on a left to right ordering of shorter to taller rectangles to signify the correctness of sorting order.

Based on this mapping comprison, we can make at least three interesting generalizations that can be observed from this mapping comparison about conceptualizations of the algorithm animation. They reflect issues of capturing the abstract functionality of an algorithm, the grain of analysis, perceptual salience, and intuitions about cultural expectations, including metaphor.

ENTITIES and their ATTRIBUTES				
Human Participants			**Algorithm Pseudocode**	**LENS**
Number Pair	**Color Pair**	**Football Pair**		
Square	Square	Stick Figure	Sorting Element	Rectangle
Number symbol	Color	Color (as weight)	Magnitude of Element	Height or Width of Rectangle
Contiguous row of squares	Non-contiguous row of squares	Contiguous row of figures	Array of Elements	Non-contiguous row of rectangles
--	--	--	Subscript of Array	--
--	--	--	Counter	--
--	--	--	Temporary Storage	--
Columns of rows (History)	Columns of rows (History)	Columns of rows (History)	--	--
--	Legend of Triangles with Color Spectrum	Legend of Color to Weight Codes	--	--

TABLE 1: Mappings from algorithm pseudocode entities and attributes to human visualizations and LENS animations

TRANSFORMATIONS				
Human Participants			Algorithm Pseudocode	LENS
Number Pair	Color Pair	Football Pair		
Color squares red	--	--	Initialize magnitudes and counter (read data)	--
Color both pink	--	Location of football	Repeat a) Reference elements to be compared	Flash elements, change fill or color
--	--	Rules of fumble, tackle	b) Compare elements same, <, >	--
Exchange numbers	Exchange colors	Change location of player or football	c) Exchange elements (swap values using temporary storage)	Exchange rectangles
Square in correct order turns green	--	--	d) Terminate pass	--
Ordering of natural numbers, all squares green	Color squares match legend	Players ordered by weight	Terminate Sorting	Rectangles ordered by increasing height or width

TABLE 2: Mappings from algorithm pseudocode transformations to human visualizations and LENS animations

Abstract Functionality

All pairs of participants developed animations which illustrate the overall functionality of the bubble sort in an abstract way. This naturally follows from the instructions given the participants that they "explain" how the sort works. If they had been given instructions to debug the code using a visualization, a very different result might have occurred.

The name "bubble" sort illustrates the abstract notion that sorting occurs by moving larger elements to the "top". This is the essence of the algorithm. All of the visualizations, including the LENS implementation, capture that notion. Observers can see the elements moving into position. It is interesting to note that the visualizations all move elements from left to right, not bottom to top as real bubbles would move. The success of these visualizations in capturing the essence of bubble sort is in the high level of abstraction which they use, ignoring many details of the actual algorithm. The primary focus of sorting algorithms is in establishing an ordered relationship between the elements. Thus the visualizations all directly represent the elements to be sorted, their relative magnitude, and their relationship in terms of ordering. Three of the visualizations picked geometric objects (squares, rectangles) and one picked stick figures as elements to be sorted. Magnitude was represented directly only in the LENS visualization. The human visualizations all used indirect coding—either color or numbers. Ordering relationships were represented spatially by the position of the elements.

All of the human visualizations used a direct mapping to the notion of a data structure array as it is usually represented in textbooks. For a one dimensional array, these textbook pictures show a row of contiguous cells containing values, usually numbers or letters. This contrasts with the LENS representation that used rectangles of different spatial extents.

Abstract functionality in the bubble sort is also captured by two abstract transformations that operate upon the elements being sorted: compare and exchange. All of the visualizations, including LENS, focus on these two transformations.

During comparison, all the visualizations referenced the compared elements using graphical features (color, the location of the football, flashing). However, only one visualization, that of the Football Pair, attempted to model

the conditional results (equal, greater than, less than) of the comparison. This was done indirectly through the rules of "tackling and fumbling." Exchanging elements was done directly by animation, ignoring the details of temporary storage while compared elements were moved.

None of the visualizations modeled the looping construct directly. Only indirectly through the overall animation does one see the notion of repetition. The interesting

events paradigm as implemented in LENS allows the specification of both compare and exchange. This contributes greatly to its success at modeling the bubble sort. Termination conditions were modeled directly by only one pair, the Number Pair, who used a change in color to highlight that elements were in their final state. All other visualizations used the observer's sense of natural ordering (LENS) or a legend (Color Pair, Football Pair) to specify the correct order of an element.

Grain of Analysis

Grain of analysis was critical for maintaining a high level of abstraction. Consequently, when looking in detail at the actual code for the algorithm, many entities and transformations were ignored by the AV designers. The housekeeping details of looping, comparing and exchanging elements were not represented. Thus, we see in Tables 1 and 2 that initialization of values, i.e. reading in data, incrementing subscripts, the counter as a representation of the size of the array, and the use of temporary storage were absent from human visualizations.

Perceptual Salience

The primary purpose of an algorithm animation as studied in this paper is in communication. We observed that human participants used two strategies to aid in their explanation: *focus of attention* and *perceptual economy*.

Focus attention of the human observer is created by the use of a significant difference in perceptual features such as motion, loudness, size, color, etc. of an object against a relatively unchanging background. Participants intuitively understood that. The Number Pair used colors to focus attention on the difference between sorted and unsorted elements and to identify the two elements being compared. The LENS visualization flashed compared objects. Its semantics also provide change color or fill. The position and motion of objects was the primary focus of attention for all visualizations during the exchange event. Size was used in the LENS animation to provided a direct recognition of sorted order of the rectangles.

Perceptual economy was demonstrated in the use of very simple geometric and stick figures, few colors, and few objects (4-5), and large differences in magnitude. This economy also helped to focus attention on the salient aspects of the algorithm.

Intuitions about Cultural Expectations

All of the AV designs use the observer's knowledge of culture to make sense of the animations. Metaphor is one of those aspects. The most striking thing about the human visualizations is the one invented by the Football Pair. This animation uses the metaphor of football play to cleverly communicate functional details of the algorithm, including decisions which come about as a result of the comparison. The location of the football shows the current state of the algorithm. The notion of the weight of the players, "tackling and fumbling" map to decisions about exchange. The Color Pair also uses metaphor with the concept of a spectrum to naturally order the elements by magnitude.

In addition to metaphor, cultural concepts of ordering are used such as the sequence of natural numbers (Number Pair) and ordering by spatial extent (LENS). Cultural notions of spatiality are also used heavily in the direction of motion. All animations moved from left to right, top to bottom, just as one reads English. It is the authors' experience that Chinese students expect the direction of motion in an animation to be from right to left. And finally, as noted above, the human visualizations all created spatial structures similar to those used in data structures texts to represent arrays.

CONCLUSIONS

Have we learned anything about (a) How people conceptualize algorithm animations in the first place; and (b) To what extent do such conceptualize accord with AV software. Clearly, both of these questions rely intimately on the extent to which our participants' homemade animations were indicative of how they actually conceptualized the algorithm. However, even if our study "loaded the dice" by asking users to explain the algorithm— not just to think about it—and by providing users with a *specific* kind of material (construction paper) that lent itself well to a *specific* kind of conceptualization (graphical), we maintain that the insights gained from this study are very useful in understanding the usability of AV software. Moreover, in the process of using the more flexible resource (the art supplies), the users came up with vastly different animations in all cases.

We have shown through a detailed analysis of the mapping of algorithm abstractions to both LENS and the human visualizations that:

- The human visualizations and LENS share a similar semantics for a high level of abstract functionality and maintain the same grain of analysis;
- Both human visualizations and LENS have a semantics for representing perceptual salience, although many of those found in the participants' visualizations are not available in LENS.
- LENS could not begin to equal the variety of representations that capture cultural expectations. These are manifested primarily in the use of metaphor which was revealed in the human visualizations.

LENS could not support the kinds of animations our participants created in their construction paper sessions. Although users were easily able to understand the LENS AV language and successfully implement the bubble sort, caution about generalization to usability for other algorithms must be observed here since bubble sort is an extremely simple algorithm.

LENS was used as a prototypical example of a new approach to using AV in instruction—in which the learner actively constructs the animation as opposed to viewing one constructed by the teacher. Our research suggests that visualizations aim at a very high level of functionality and thus do not provide the detailed procedural information of

algorithm pseudocode. This might account for the prior findings of Stasko et al. [4] that AV provides no advantage over text-only teaching material. Our research also deomonstrates that human visualization of algorithms can be quite creative and can vary significantly from one group to another. An interactive AV system by necessity has to predefine the underlying semantics and graphic building blocks of the animation language. This type of software design begs for a close match between what humans conceptualize and what the AV language provides. We wonder if negative empirical findings regarding the utility of AV reflect a mismatch between human conceptualization and AV representation. We recommend more detailed studies on how humans conceptualize, visualize and implement algorithms to develop true usability of AV systems.

ACKNOWLEDGMENTS

We gratefully acknowledge John Stasko and his colleagues for allowing us to use a prototype version of LENS, and for graciously and patiently accepting our criticisms regarding its usability.

REFERENCES

1. Mukherjea, S., & Stasko, J.T. Applying algorithm animation techniques for program tracing, debugging, and understanding. *Proc. 15th IEEE International Conference on Software Engineering* (Baltimore, MD, 1993), pp. 456–465.

2. Van Dam, A. The electronic classroom: Workstations for teaching. *International Journal of Man-Machine Studies 21* , 4 (1984), 353–363.

3. Brown, M. *Algorithm Animation* . Cambridge, MA: The MIT Press, 1987.

4. Stasko, J., Badre, A., & Lewis, C. Do algorithm animations assist learning? An empirical study and analysis. *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems* (Amsterdam, The Netherlands, 1993), pp. 61–66.

5. Stasko, J.T. Tango: A framework and system for algorithm animation. *IEEE Computer* (September 1990), pp. 27-39.

6. Douglas, S.A. and Liu, Z-Y. Qualitative simulation and causal explanation in an intelligent tutor. *Proceedings of the International Conference on Artificial Intelligence*, (Detroit, MI, August, 1989).

7. Baecker, R.M, & Sherman, R.M. *Sorting out sorting.* 16mm color sound film shown at SIGGRAPH '81 (Dallas, TX, 1981).

8. Paivio, A. *Mental representations: A dual coding approach.* New York: Oxford University Press, 1990.

9. Miyake, N. Constructive interaction and the iterative process of understanding. *Cognitive Science 10,* (1986), 151–177.

10. Suchman, L. *Plans and situated actions: The problem of human–machine communication.* Cambridge: Cambridge University Press, 1987.

11. Douglas, S.A. Conversational analysis and human-computer interaction design. P.Thomas (Ed.) *The Social and Interactional Dimensions of Human-Computer Interfaces* Cambridge University Press, (in press).

12. Naps, T.L, & Hundhausen, C.D. The evolution of an algorithm visualization system. *Proc. 24th Annual Small college Computing Symposium* (Morris, MN, 1991), pp. 259-263.