Dancing Hamsters and Marble Statues: Characterizing Student Visualizations of Algorithms

Teresa Hübscher-Younger[*]
Language, Literature & Communication
Rensselaer Polytechnic Institute
Troy, NY 12180

N. Hari Narayanan[‡]
Computer Science & Software Engineering
Auburn University
Auburn, AL 36849

Abstract

Algorithm visualization research for computer science education has primarily focused on expert-created visualizations. However, constructionist and situated theories of learning suggest that students should develop and share their own diverse understandings of a concept for deep learning. This paper presents a novel approach to algorithm learning by visualization construction, sharing, and evaluation. Three empirical studies in which students engaged in these activities are discussed. The resulting learning benefits are quantified, and student visualizations are characterized in multiple ways. Then another study that investigated how specific characteristics of such visualizations influence learning is described. This work demonstrates the effectiveness of having students create algorithm visualizations, identifies characteristics of student-created algorithm visualizations and illuminates the learning benefits derived from these characteristics.

CR Categories: K.3.1 [Computers & Education]: Computer Uses in Education – Collaborative Learning; K.3.2 [Computers & Education]: Computer & Information Science Education - Computer Science Education, Self-assessment.

Keywords: algorithms, empirical studies, learning, multimedia, representation, visualization.

1 Introduction

As a recent survey [Hundhausen et al. 2002] indicates, a predominant portion of the literature on algorithm visualization from the last two decades focuses on expert-created visualizations intended for a student audience. However, constructionist and situated theories of learning [e.g., Lave & Wenger 1991; Linn & Burbules 1993] hold that learning will be deeper if mediated by active creation and negotiation by individual and communities of learners. These theories suggest that students must develop and share their own diverse understandings of a concept. This presents a dilemma to those interested in using or researching algorithm visualizations as learning tools: if students gain their understanding solely from expert-created materials, how likely are they to develop a diverse set of insights?

----

[*]e-mail: hubsct@rpi.edu
[‡]e-mail: narayan@eng.auburn.edu

Meanwhile, perhaps spurred by the lack of strong empirical proof that such expert-created algorithm visualizations significantly improve student learning, a number of researchers have of late investigated an alternative pedagogy: students learning by constructing their own algorithm visualizations [e.g., Anderson & Naps 2001; Astrachan & Rodger 1998; Hundhausen & Douglas 2000; Stasko 1997]. Several questions can be raised about such a constructive approach to learning from algorithm visualizations. Do students exhibit significant improvements in their understanding of algorithms as a result of building their own visualizations? What is the nature of student-created algorithm visualizations? How do characteristics of student-created visualizations affect their learning?

Extant literature on algorithm visualization is insufficient to provide unequivocal answers to these questions. Survey, observational and anecdotal data [Astrachan & Rodger 1998; Stasko 1997] do suggest that students are motivated by, and benefit from, constructing their own visualizations. However, it is not clear whether these benefits are of a statistically significant or generalizable nature. Hundhausen and Douglas [2000] investigated the comparative question, and found no performance differences between a group of students that interacted with an expert-created animation and another that constructed their own animations using art supplies. We are not aware of any studies that have rigorously explored the nature and learning effects of student-created algorithm visualizations

It is in this context that we present new research on student-created algorithm visualizations in this paper. We use the term algorithm visualization to mean a multimodal (i.e. visual and verbal) representation that could involve a variety of media (i.e. sound, animation, graphics or text), and which helps students internally visualize the operations of an algorithm. First, we present a novel approach to algorithm learning by visualization construction, sharing, and peer-evaluation, and outline a system designed to support students in these activities. Second, we provide an overview of three empirical studies in which we had students create algorithm visualizations and evaluate the visualizations created by their peers on six characteristics: familiarity, originality, pleasure, salience, understandability, and usefulness. The resulting learning benefits are quantified, and the visualizations are further characterized in terms of media use, presence of metaphors and similarity to familiar styles. We then discuss a study that considered how specific characteristics of such visualizations influence learning. Finally, the paper concludes with a summary of the contributions of this research. Related work is discussed throughout the paper as appropriate.

2 Our Approach

In order to learn an algorithm, we have students go through an assignment consisting of the following four stages in the given order.

1. *Individual Construction*: Construct a visualization of the algorithm, given its pseudocode. We intentionally keep this open-ended, in that students are free to use any technique and software tools they desire; in particular, no algorithm animation creation tools are prescribed, and instructions to students used the more neutral term "representation" instead of visualization to avoid bias toward graphical representations and animations.

2. *Sharing with the Community*: Exhibit the visualization to other students in the class.

3. *Peer Evaluation*: Rate visualizations created by everyone else using a five point Likert scale along the following six characteristics that we developed in order to characterize student-constructed visualizations.

- *Familiarity* ("How familiar were you with the content of the representation?")
- *Originality* ("How much did this representation differ from the other representations?")
- *Pleasure* ("How much did you enjoy the way this representation communicated the algorithm?")
- *Salience* ("How well did this representation point out the important features of the algorithm?")
- *Understandability* ("How easy was this representation to understand?")
- *Usefulness* ("How central was this representation to your understanding of the algorithm?")

4. *Collaborative Discussion*: Commenting on others' visualizations and responding to comments on one's own visualization.

This approach was developed based on several considerations. First, evidence in the literature [Chi et al. 1994] points to beneficial effects of individually trying to understand an unfamiliar concept in order to explain it, to oneself or others. This motivated the first stage of individually constructing an explanatory visualization in an unrestricted manner.

In addition to individual reflection, group interaction and negotiation to collaboratively build knowledge and shared understanding are also important to learning within a community. This suggests that students need to work together as well as alone. Indeed, in earlier studies we found that computer science undergraduate students already form a community of practice in which learners tend to study collaboratively, discussing and constructing a shared understanding of topics covered by their instructor [Hübscher-Younger & Narayanan 2001; 2002]. Unfortunately, we also found that authority assigned to certain explanatory visualizations (typically those provided by the instructor) led students to ignore alternative explanations and fail to understand the limitations of such visualizations [Hübscher-Younger & Narayanan 2002]. We felt that visualizations built by students themselves carry less authority. This would encourage students to consider multiple peer-constructed visualizations instead of a few expert-provided ones, examine a particular visualization's limitations as well as advantages, evaluate it accordingly, and engage in discussions. This was the motivation behind stages 2, 3 and 4.

A web-based system called CAROUSEL (Collaborative Algorithm Representations Of Undergraduates for Self-Enhanced Learning) was designed to support students in the second, third and fourth stages. CAROUSEL is implemented using MySQL, PHP, Javascript and HTML. A MySQL database running on an Apache server stores visualizations that students upload, evaluation information, comments, responses, user profile information and student grades for activities. The server presents students' visualizations on web pages dynamically generated using PHP. The web-based interface allows students and instructors to set up course, schedule and activity information and to submit visualizations, evaluations, comments, responses and grades. Once all visualizations for a particular algorithm have been uploaded, the instructor (or experimenter) can set a date on which they will become publicly, but anonymously, available for viewing, evaluating and commenting. CAROUSEL has additional features for the instructor of a course to help manage course activities. It permits students to share visualizations, provides constant access to the visualizations, and allows the instructor/experimenter to implement scheduled stages of assignments. See [Hübscher-Younger & Narayanan 2003a] for additional information on this system.

3 Efficacy Of Visualization Construction

In this section we summarize results from three studies designed to investigate whether the visualization construction activity leads to improved understanding of algorithms and what types of visualizations students create. All studies had a common structure. Student volunteers from a data structure or algorithms course were solicited to participate for extra credit. In each study several algorithms were assigned, and participants were asked to construct, share, evaluate and discuss visualizations of these. For each algorithm, participants first took a pretest to measure their baseline knowledge about the algorithm. Then the algorithm was assigned by making its pseudocode available through CAROUSEL. After this, participants generally had one week to construct and upload a visualization. We placed no restrictions on the nature of, or media to be used for, these visualizations. We also called these "representations" instead of "visualizations" to avoid any bias toward graphics and animation. Students were free to create these with paper, in which case scanning services were offered to digitize such submissions. After the deadline for uploading visualizations was over, CAROUSEL made these publicly available to all participants. They were then given about a week to rate every visualization (other than their own) on the aforementioned six characteristics and to input comments and responses. Afterwards they were given a posttest.

The pretests and posttests included three kinds of questions: conceptual questions that probed factual knowledge (e.g. what is the element used to partition an array in quick sort called?), procedural questions that probed the understanding of how the algorithm manipulates data (e.g., given an array, show the array's contents at the end of the third recursive call to quick sort), and simulative questions that probed the ability to mentally simulate the algorithm on given data (e.g., draw a recursion tree showing all recursive calls that will be made for a given initial input). The four stages generally overlapped, so that students were creating and sharing new visualizations as well as rating and commenting on old visualizations in any given week. Extra credit was attached to each stage of an assignment, and students could choose which stages they completed. The minimum requirement was to take the pretest and the posttest, but a participant could choose whether to create a visualization, evaluate others' visualizations and engage in discussion.

Student learning performance (the difference between posttest and pretest scores) was evaluated and compared based on which activities they chose to do, i.e. whether they had chosen to both create representations and evaluate others' representations or to just evaluate representations. All the students who created a representation for a particular assignment did evaluate all the other representations, although they could have chosen not to do this. Also, no student chose to just take the pretest and posttest without evaluating any representation, another possible choice.

3.1 The First Study

Twelve students volunteered from a data structures and algorithms course taken mostly by sophomore students. The study was conducted over four weeks with three algorithm assignments: calculating the Fibonacci series, selection sort and merge sort. The study did not use a pretest, and one rating characteristic used was *contiguity* ("how well did this representation connect with the other representations for this algorithm?") instead of *originality*. CAROUSEL identified authors when publicly displaying the visualizations. Also, the threaded discussion feature of CAROUSEL was not ready at the time of this study, so students were asked to do only creation, sharing and evaluation. A total of 36 visualizations were created by 11 of the 12 volunteers. For two of the three algorithms that were used in this study (selection and merge sort), there was a significant positive correlation between creating a visualization and posttest scores (r=.635, p=.07; r=.663, p=.05), suggesting that visualization creation improves learning.

After this study, contiguity was replaced with originality, a discussion capability was added to CAROUSEL, and the system was changed to hide author information during the sharing, evaluation and commenting stages. The rating characteristic was changed and authors made anonymous in order to discourage convergence of representations, but a detailed discussion of this issue is beyond the scope of this paper. See [Hübscher-Younger & Narayanan 2003b] for details.

3.2 The Second Study

Table 1. Test Scores

Algorithm	Pretest	Posttest	Learning
Fibonacci	42%	59%	17%
Exponentiation	14%	51%	37%
Binary Search Tree	33%	62%	29%
Leftist Heap	12%	31%	19%
Selection Sort	40%	66%	26%
Merge Sort	11%	41%	30%
Quick Sort	29%	55%	26%
Disjoint Set	15%	56%	41%
Depth First Search	19%	50%	31%

Sixty students in an introductory algorithm analysis course (this course is generally taken by juniors, and a few seniors and beginning graduate students) participated in this study. The study was conducted over 12 weeks with nine algorithm assignments: calculating the Fibonacci series, exponentiation, inserting a node in a binary search tree, merging leftist heaps, selection sort, merge sort, quick sort, find with path compression in a disjoint set and depth-first search. Each assignment included taking a pretest, the four stages with six rating characteristics described in Section 2, and taking a posttest. A total of 196 visualizations were created by 36 of the 60 volunteers.

Overall results implied that the activities of creating, sharing and evaluating algorithm visualizations aided learning. The student participants improved their score from pretest to posttest by 30% on average across all algorithms. Table 1 displays the average normalized pretest, posttest and learning scores for each algorithm. Learning was measured as the difference between the pretest and posttest scores for each algorithm. Scores were normalized by dividing the raw score by the maximum score attained by any student on a test for a particular algorithm.

Moreover, visualization construction had a clear beneficial effect. There was a significant difference between the normalized posttest scores (F(1,327)=14.4, p<.001) and the normalized learning scores (F(1,327)=3.63, p=.058) of those who had and those who had not created visualizations. In both cases, students who created visualizations had higher scores than those who did not (i.e. those who only evaluated and commented on others' visualizations). The mean for the normalized posttest scores for the students who did not create a visualization was 46% and for the students who did create one was 57%. The mean for the normalized learning scores for the students who did not create a visualization was 25% and for those who did create one was 31%. When multiple linear regression analysis techniques were used to look at how visualization creation and the algorithm covered affected normalized learning scores, it was found that the model was significant (F(9,318)=3.37, p<.001) and creating a visualization had a significant positive effect on learning when the choice of algorithm was controlled for (F(1,318)=5.025, p=.026).

In particular, the learning scores were significantly related to whether a student created a visualization for two of the nine algorithms: the recursive exponentiation algorithm (F(1, 35) =5.06, p=.03) and quick sort (F (1,34) =3.99, p=.05). In the case of the recursive exponentiation algorithm, the students who did not create a visualization increased their score by 27%, and the students who did create one increased their score by 45%. In the case of quick sort, the students who did not create a visualization increased their score by 17%, and the students who did create one increased their score by 34%.

3.3 The Third Study

Table 2. Test Scores

Algorithm	Pretest	Posttest	Learning
Disjoint Set	20%	63%	43%
Shortest Path	42%	67%	25%
Huffman's Codes	8%	56%	48%
Depth First Search	21%	42%	21%

In this study 43 students in the same algorithms course (but in a different term) created visualizations of four algorithms (find with path compression in a disjoint set, Dijkstra's shortest path, Huffman's codes and depth-first search) over a four-week period. Each assignment included pretesting, the four stages with six rating characteristics described in Section 2, and posttesting. The schedule was quite compressed in this study due to circumstances beyond our control, so students were dealing with multiple algorithms during each week. A total of 65 visualizations were created by 22 of the 43 volunteers.

Overall results implied that the activities of creating, sharing and evaluating algorithm visualizations aided learning. The student participants improved their score from pretest to posttest by 40% on average across all algorithms. Table 2 displays the average normalized pretest, posttest and learning scores for each algorithm. Learning was measured as the difference between the pretest and posttest scores for each algorithm. Scores were normalized by dividing the raw score by the maximum score attained by any student on a test for a particular algorithm.

Furthermore, visualization construction had a clear beneficial effect. The normalized posttest scores (F(1,94)=5.44, p=.02) and the normalized learning scores (F(1,94)=4.43, p=.04) across all algorithms were significantly different between those students who created visualizations and those who only evaluated and commented on others' visualizations. In both cases, the students who created the visualizations had higher scores than those who did not. The mean for the normalized posttest scores for the students who did not create a visualization was 48.2% and for the students who did create one was 62%. The mean for the normalized learning scores for the students who did not create a visualization was 26.3% and for those who did create one was 40.2%. When multiple linear regression analysis techniques were used to analyze how visualization creation and the algorithm covered affected normalized learning scores, it was found that the model was significant (F(4, 90)=5.71, p<.001). In the model, creating a visualization had a significant positive effect on learning (F(1,90)=7.351, p=.008).

In particular, the normalized learning scores were significantly related to whether a student created a visualization for the algorithm to generate Huffman's Codes (F (1, 23)=5.92, p=.02). The average normalized learning score for this algorithm was 57% for those students who did create visualizations and 28% for those who did not.

3.4 Discussion

It is fairly well established in the general education literature that constructing one's own explanations of a concept can lead to better learning. But such work [e.g., Chi et al. 1994] generally pertains to children learning concepts. Our studies provide evidence that this holds true for adult college students learning a complex subject by constructing explanatory visualizations. These results add to similar evidence presented by Hundhausen and Douglas [2000]. What sets these studies apart, however, is that this is the first time to our knowledge that a systematic series of studies have demonstrated the benefits of visualization construction in the algorithm domain with quantitative data. Our studies were conducted over 20 weeks during three terms, and involved over 100 students working with 11 different algorithms.

Hundhausen [2002] reports on ethnographic studies of students constructing visualizations, but the students were encouraged to build "animations" and "visualization storyboards", biasing them toward a certain kind of visualizations. What our studies showed was that construction of explanatory visualizations, *unfettered* by both specific instructions and the need to learn a non-trivial algorithm visualization language, does lead to improved learning. This is an important result, given the time and effort that researchers expend on building newer and better algorithm animation tools [e.g., Lahtinen et al. 1998], and the time and effort that an instructor and students of an algorithms course have to set aside for learning to use such tools. For instance, Hundhausen and Douglas [2002] report that in one particular study they found students spending 33 hours on average constructing an algorithm visualization, most of which was devoted to low-level graphics programming needed to make animations work.

Even the use of so-called low-fidelity algorithm visualizations [Hundhausen & Douglas 2002] introduces a layer of complexity over and above that of understanding the underlying algorithm: in this case translating the algorithm's mathematical logic to the spatial logic of a scripting language. As an example, a sample script for bubble sort provided in the aforementioned paper is 42 lines long, whereas the pseudocode for this algorithm can be written in about 8 lines. End-user programming languages like SALSA that these authors propose for building low-fidelity algorithm visualizations may ameliorate this translation problem somewhat compared to the graphical programming required with high-fidelity algorithm visualization construction tools. While it can be argued that representational translation is unavoidable, and perhaps even desirable, in learning, specific algorithm visualization construction tools force certain kinds of translations. Any such tool will require students to spend time learning it, and bias them toward a particular kind of visualization – one that involves algorithm animations. But recent research [Narayanan & Hegarty 2002; Tversky et al. 2002] calls into question the communicative efficacy and educational benefits of animations and interactivity found in typical computer-based visualizations.

So our studies explored a more radical position: why not have students define a visualization in whatever way they want (thereby defining their own representational translations), and construct it using whichever tools they have at their disposal? Visualizations in our studies were constructed by students using a variety of tools – simple text and html editors, Javascript, Macromedia Flash, etc., and many did not contain any graphical representations whatsoever. Whether students will learn more or less if they construct specific kinds of visualizations using algorithm animation construction tools is a question yet to be addressed. The studies reported here provide baseline data on unrestricted self-construction of algorithm visualizations, a prerequisite to future comparative studies investigating such questions.

One possible explanation for the increased learning evidenced in these studies is that since self-selection was involved, only the brightest and most motivated students constructed visualizations. However, informal feedback indicated that several struggling students participated in the hopes of improving their course grades, since extra credit was awarded for participation and was not dependent on pretest and posttest scores. Furthermore, 92%, 60% and 51% of those who participated constructed at least one visualization in each study. These suggest that a broad group of students, not just the smart and motivated ones, participated, and that a majority of participants engaged in visualization construction.

There were three factors that we did not control for. Even though we avoided teaching an algorithm in class at the same time it was being used in an assignment, given the nature of the study it was impossible to control for students learning about an algorithm outside the assigned tasks. Similarly, volunteers in the studies presumably spent more time studying algorithms than those who did not participate. But since most of the students in the classes in which these studies were conducted chose to participate, and since all of the volunteers undertook at least \representation evaluation, we did not have control groups that did not use CAROUSEL but who spent a similar amount of time doing other educational activities with these algorithms for comparison purposes.

4 Characterizing Visualizations

Now we turn to a discussion of characterizing the visualizations constructed by students in these studies. Participants rated each visualization along six characteristics: familiarity, originality, pleasure, salience, understandability and usefulness. These raw ratings and their average values provide a set of peer-measures for the perceived quality of a visualization. The experimenter did three additional analyses on the visualizations. The first was to characterize *media use*. Media use was rated on a scale of 1 to 4 with level 1 being the use of only text, level 2 being the use of graphics and/or text, level 3 being the use of graphics, text, sound and/or animation and level 4 being the use of all of the previously mentioned plus hypermedia and/or 3-D animation. The second was to characterize the *metaphoric content* of a visualization. Metaphoric content was rated with dummy variables to indicate whether metaphors beyond the kind conventionally found in textbooks were present or not. The third was to characterize how *similar* a visualization was to explanations found in the course textbook or lectures. The experimenter rated visualizations on a scale of 1 to 5 with 1 being a rating for visualizations that are least like a textbook or classroom explanation and 5 being a rating for visualizations that are most like a textbook or classroom explanation.

4.1 Media Use

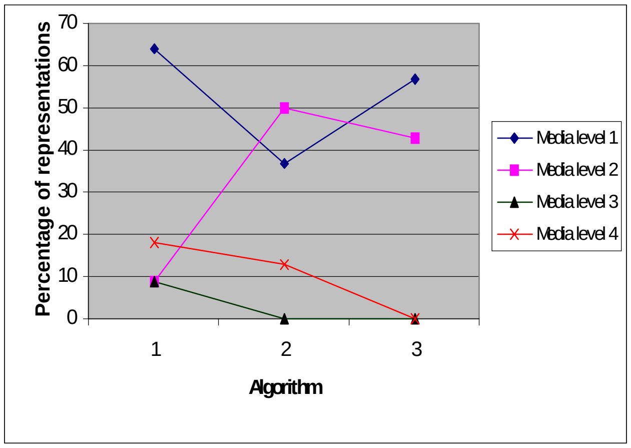


Figure 1. Media use patterns in study 1.

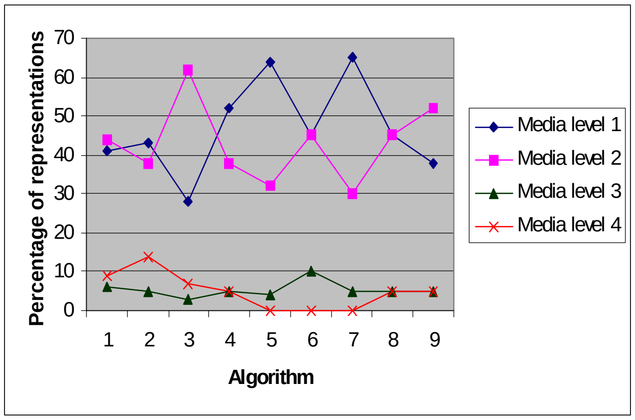


Figure 2. Media use patterns in study 2.

At the beginning of the first study, students chose to work with a wide variety of media, including text, graphics, sound and animation, based on their personal preferences. However, over the course of the study, the students converged on a simple style, one incorporating primarily graphics and text (Figure 1).

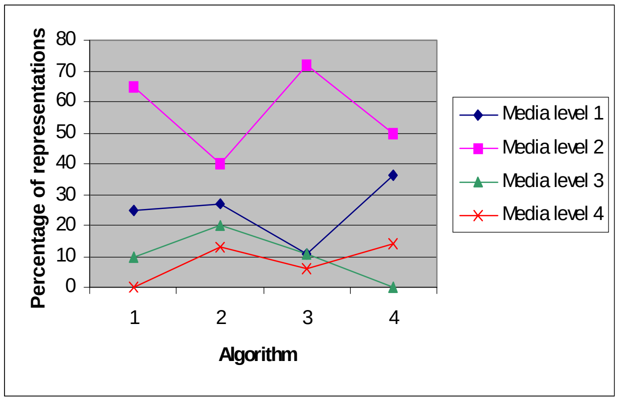


Figure 3. Media use patterns in study 3.

It was found in the first study that adding graphics, animation and sound to text resulted in a higher overall rating. However, when more complicated media, such as hypermedia or 3-D animation were added, the ratings did not rise. Multiple linear regression analysis techniques were used to explore how the use of different media affected student ratings. The parameter estimates for media use led to consistent conclusions about how media use affected student ratings. The effect of adding graphics to text was positive in all cases, except for the familiarity rating, where only a small negative effect was estimated (-0.02). The effect of adding animation or sound was always positive, except for the contiguity rating, where only a small negative effect was estimated (-0.02). Adding hypermedia and/or 3-D animation always led to a large significant negative effect on the rating.

Student ratings of all visualization characteristics were significantly affected ($p<.0001$) by the type of media used in the second study. The results are similar to the first study. Adding graphics to text improved the rating of all characteristics. Adding sound and/or animation to a visualization improved the pleasure and originality ratings more than 0.5 points and the understandability rating 0.1 point. Adding hypermedia (there were actually no examples of 3-D animation in this study) improved the ratings of all characteristics, except pleasure and originality. Figure 2 shows levels of media use in visualizations across algorithms in the second study.

Media use in study three is illustrated in Figure 3. Adding graphics to text always led to increased ratings. Adding sound and/or animation always led to higher ratings for pleasure and originality, and sometimes led to other rating benefits. Adding hypermedia or interactive components did not necessarily lead to increased ratings and sometimes led to decreased ratings. In studies two and three, media use converged less toward any specific style compared to the first study.

4.2 Metaphoric Content

In the first study it was found, using multiple logistic regression analysis techniques, that the existence of unconventional metaphoric content had a significant effect on the student ratings for salience and pleasure, $G^2s(5, 236) = 9.7$ and 14.5, $ps<0.1$ and 0.05, respectively. The lack of metaphoric content led to higher salience ratings, and the presence of metaphoric content led to higher pleasure ratings.

The existence of a metaphor outside those normally used in the classroom was significantly related to the student ratings of pleasure and originality (F(1, 205)= 8.37 and 23.38, p<.005, respectively) in the second study. In both cases the relation was positive, so the use of an unconventional metaphor led students to rate the visualization more enjoyable and original.

In the third study, unconventional metaphor was significantly and negatively related to the student ratings of all characteristics (p<.0001). However, unlike the second study, only a small portion (15%) of the representations in this study contained an unconventional metaphor.

4.3 Similarity to Familiar Styles

Similarity (to textbook or classroom styles) ratings of visualizations from the first study increased over time with each new algorithm: the first algorithm had an average rating of 3.4; the second had 3.9; and the third had 4.7. Furthermore, the average of all the ratings the students gave each visualization was significantly positively related to the rating of how similar that visualization was to a textbook or classroom explanation (F(1, 24)=3.9, p=.06). Multiple linear regression analysis techniques were used to explore how the ratings of the visualizations' similarity to textbook or classroom explanations were related to the student ratings of different characteristics. The similarity rating's relations to students' ratings of usefulness, salience and contiguity were positive and significant (F(1,24)=6.5, 6.0, and 10.6 respectively, p<.05). In other words, how similar a visualization was to a textbook or classroom explanation positively influenced not only the overall rating that visualization received, but also student ratings for how useful that visualization was for their understanding of the algorithm; how well that visualization pointed out the salient features of the algorithm; and how well it was contiguous with (built upon) the other visualizations for that algorithm. For understandability and familiarity, the effect was also positive, but not significant. Interestingly, pleasure was the only student rating that was negatively affected by a visualization's similarity to a textbook or classroom explanation, but the effect was not significant.

Unlike in the first study, there was not a trend toward the visualizations becoming more similar to those found in the textbook in the second study. The average of all the ratings the students gave each visualization was not significantly positively related to the rating of how similar that visualization was to a textbook or classroom explanation. Multiple linear regression analysis techniques were used to explore how the ratings of the visualizations' similarity to textbook or classroom explanations were related to the average student ratings of different characteristics. The similarity ratings' relation to students' ratings of usefulness, understandability and salience were positive and significant (F(1,205)= 4.56, 5.25 and 9.65 respectively, p<.05). In other words, student ratings of how useful a visualization was, how understandable it was, and how well it pointed out the important points of the algorithm were positively influenced by how similar that visualization was to classroom and textbook conventions. Familiarity was positively related, but the effect was not significant. The similarity ratings' relation to students' ratings of pleasure and originality were negative and significant (F(1,205)= 4.09 and 16.6 respectively, p<.05). In other words, students rated a visualization higher in pleasure and originality, if it differed from their classroom conventions.

There was not a trend toward the visualizations becoming more like classroom or textbook explanations in study three. The average of all the ratings the students gave each visualization was not significantly related to the rating of how similar that visualization was to a textbook or classroom explanation. Multiple linear regression analysis techniques were used to explore how the ratings of the visualizations' similarity to textbook or classroom explanations were related to the average student ratings of different characteristics. The similarity ratings' relation to students' rating of originality was significant, but the relation was negative. In other words, the higher the similarity rating was, the lower the students rated the originality of the visualization.

4.4 Diversity

Students constructed a variety of visualizations, differing both in style and content. Some produced entertaining animations that illustrated the mathematical basis of an algorithm, such as the Dancing Hamsters showing the Fibonacci series (Figure 5). Others submitted pure text containing elaborate metaphorical stories that illustrated what an algorithm computes (but not how), such as a story about a shopkeeper who used the Fibonacci series to balance his shelf of statues (Figure 6). A large portion of the visualizations were in a walkthrough style (60% in the second study and 70% in the third), giving an example of a data set and showing how it would change over time as the algorithm operated on it. This style focused on the execution of pseudocode. There were also graphical representations of pseudocode (Figure 7), interactive calculators that illustrated algorithm efficiency, textual representations that explained the main ideas and reasoning behind an algorithm, and visualizations that only represented the results of algorithm execution on a specific input data set.

5 Linking Visualization Characteristics to Learning Benefits

Now we turn to a study that addressed the following question. How do specific characteristics of student-created visualizations affect what other students can learn about the algorithm from these visualizations (when the pseudocode is also provided)?

While analyzing students' visualizations we discovered that salience and pleasure were important characteristics, because the representations differed most in these characteristics. Visualizations with unconventional metaphoric content generally scored higher in pleasure and lower in salience, and visualizations using a style of explanation similar to the course textbook (which generally used a walkthrough style) had in general higher salience scores, but lower pleasure scores. If a student considers a particular style of explanation more pleasurable than others, he or she is likely to be more engaged with that visualization, and hence learn more from it. Salience ("how well did this representation point out the important features of the algorithm?") indicates the extent to which a particular visualization captures all the important features of an algorithm. So the higher the salience rating of a visualization, the more one is likely to learn from it. In order to explore this issue further, we conducted the following study.

5.1 Procedure

Forty-five students enrolled in an introductory algorithm analysis class participated for extra credit. They were split into two matched (based on course standing) groups: Group 1 (23 students) and Group 2 (22 students). Participants met with the experimenter in a computer lab. They were given an introduction to the study and then asked to take a pretest. After completing the pretest, the students were given the pseudocode for three algorithms (Fibonacci, exponentiation and binary tree insertion) on paper, a posttest and a URL to access a website. The URLs given to each participant presented different visualizations based on the student's assigned group, as explained below. The students could work with the pseudocode and visualizations to answer the posttest questions, taking as much time as they needed. Learning was measured as the difference between the score on the pretest and the score on the posttest. Students could look at the pseudocode and visualizations while taking the posttest.

The URL for Group 1 contained three visualizations. One was for the Fibonacci algorithm that received the highest average salience rating (4.32) among all visualizations of this algorithm in the three previous studies. The second was for the exponentiation algorithm that received the highest average pleasure rating (3.55) among all visualizations of this algorithm in the three previous studies. The third was for the binary search tree insertion algorithm that had the maximum positive (average pleasure rating – average salience rating) value (0.61) among all visualizations of this algorithm in the three previous studies. The URL for Group 2 contained three visualizations. One was for the Fibonacci algorithm that received the highest average pleasure rating (4.11) among all visualizations of this algorithm in the three previous studies. The second was for the exponentiation algorithm that received the highest average salience rating (3.84) among all visualizations of this algorithm in the three previous studies. The third was for the binary search tree insertion algorithm that had the maximum positive (average salience rating – average pleasure rating) value (0.59) among all visualizations of this algorithm in the three previous studies.

The two visualizations with the highest average pleasure ratings that the groups saw were both created by the same person and had a similar style. They were both humorous stories using only text. The one for the Fibonacci series was titled "A Tale of Fear, Loathing and Greed on a College Campus" and was about how a university employee used the algorithm to determine how many tickets to give out each day. The one for the exponentiation algorithm titled "A Tale of Two Engineers" told the story of engineers from two rival universities writing iterative and recursive algorithms for exponentiation.

The two visualizations with the highest average salience ratings that the groups saw were also created by the same person and had a similar style. They both had a discussion of the problem, a comparison between iterative and recursive algorithms for solving the problem, and interactive components that calculated results so that students could see the efficiency difference between the two discussed implementations.

The binary tree insertion visualizations had different authors. Group 1's visualization contained an animation while Group 2's visualization had text and static graphics.

5.2 Results

Table 3. Test Scores

Algorithm	Pretest	Posttest	Learning
Fibonacci	39%	75%	36%
Exponentiation	20%	69%	49%
Binary Search Tree	56%	71%	15%

The maximum score possible on the pretests and posttests for the different algorithms differed. Although there was one pretest and one posttest given to the students, the tests were separated into three sections, one for each algorithm. For the purposes of analysis, the three sections are treated as three different tests. The maximum possible score for the Fibonacci algorithm was 18, the maximum score for the exponentiation algorithm was 11, and for the binary search tree insertion algorithm was 12. Test scores were normalized to percentages of the appropriate maximum. Table 3 provides the pretest, posttest and learning score percentages for each algorithm averaged over the entire set of 45 participants. Clearly, students were able to learn the algorithms from pseudocode with the help of peer-constructed explanatory visualizations.

Multiple linear regression analysis techniques were used to see how the learning scores associated with a visualization were related to its average rating of salience, average rating of pleasure and the algorithm covered. The whole model with these variables was significant (F(4, 130)=13.1, p<.0001). The learning score's relations to the visualization's average rating of salience and average rating of pleasure were both positive and significant (F(1,130)=5.8 and 4.8, p=.02 and .03 respectively). In other words, the higher the salience rating and the higher the pleasure rating of a visualization, the more a student learned from it. The parameter estimates for the average salience and pleasure ratings are 22.1 and 39.1 respectively, which suggest that how pleasurable a visualization has more of an effect than the visualization's salience on how much a student is able to learn from it.

This result was also obtained when a similar analysis that included all six characteristics was performed. A multiple linear regression analysis technique was used to see how the learning scores associated with a visualization were related to its average ratings of salience, pleasure, originality/familiarity and usefulness/understandability. Originality and familiarity (also usefulness and understandability) were combined for this analysis because these two sets of characteristics turned out to be highly correlated in the sample of visualizations used. The whole model with these variables was significant (F(4, 130)=6.02, p=.0001). The learning score's relations to the visualization's average ratings of salience and pleasure were again both positive and significant (F(1,130)=10.37 and 3.29, p=.002 and .072 respectively).

5.3 Discussion

The interesting finding from this study is that pleasure and salience are characteristics of student-created visualizations with a positive impact on how much other students gain from these visualizations. The implication of this for the design of algorithm visualizations is that making them pleasurable to use is not just icing on the cake, but has an impact on students' ability to derive information from the presented materials. It is also important, but not enough, to make the central points regarding the algorithm stand out well in the visualization. Also note that the highest pleasure ratings for two algorithms went to visualizations that only involved text, the contents of which were humorous and related to everyday experiences of the student community. It is often believed that the more polished the graphics or the more advanced the media, the more pleasurable students will find the presentation. When students become designers and evaluators of visualizations themselves, they do not conform to this presupposition.

6 Conclusion

Constructionism is gaining hold in the research on algorithm visualization for education. Computer support for visualization construction typically comes in the form of a custom-designed scripting or programming language and a system that interprets the language to generate various kinds of visualizations. But what if computer support is provided *instead* in the form of a system that allows students to exhibit and evaluate their visualizations, which they construct with whatever means they have at their disposal? Will they learn from these activities? What kinds of algorithm visualizations will they construct? Are there characteristics of visualizations that are predictors of how much viewers of these visualizations will learn from them? This paper presented studies and analyses addressing these questions.

We found that authoring and evaluating visualizations helped students learn about algorithms. Moreover, those who authored visualizations learned significantly more than those who only evaluated others' visualizations did. Students constructed a richer and more diverse set of visualizations than those found in typical instructional materials. They also rated visualizations of their peers on six characteristics. Contrary to the conventional wisdom that multimedia is attractive to young adults, students did not give high ratings to visualizations with complex media. We also found that viewers tend to learn more from visualizations that were rated high in salience and pleasure.

The first contribution of this work is to provide quantitative evidence to show that novices constructing, sharing and evaluating algorithm visualizations do indeed learn about the underlying algorithms from these activities. These are activities that can be instituted in the classroom with minimal effort on the part of an instructor and minimal training on the part of students. They can be supported with a simple web-based system that allows students to exhibit, rate and discuss the visualizations. The second contribution is that it presents multiple ways of analyzing student constructed algorithm visualizations: in terms of media use, metaphoric content and similarity to conventional styles, or rated on a set of six characteristics. Two of these, salience and pleasure, were found to positively impact learning.

7 Acknowledgments

Studies reported here form a part of the first author's unpublished doctoral dissertation [Hübscher-Younger 2002]. This research was supported by NSF under contract REC-9815016.

References

ANDERSON, J. M., AND NAPS, T. L. 2001. A context for the assessment of algorithm visualization systems as pedagogical tools. In *Proceedings of the First International Program Visualization Workshop*, University of Jonesuu Press, 121–130.

ASTRACHAN, O., AND RODGER, S. H. 1998. Animation, visualization and interaction in CS 1 assignments. In *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education*, ACM Press, 317–321.

CHI, M. T. H., DE LEEUW, N., CHIU, M.-H., AND LAVANCHER, C. 1994. Eliciting self-explanations improves understanding. *Cognitive Science 18*, 3, 439–477.

HÜBSCHER-YOUNGER, T. 2002. *Understanding Algorithms through Shared Representations*. PhD thesis, Auburn University.

HÜBSCHER-YOUNGER, T., AND NARAYANAN, N. H. 2001. How undergraduate students' learning strategy and culture effect algorithm animation use and interpretation. In *Proceedings of the First IEEE International Conference on Advanced Learning Technologies*, IEEE Press, 113–116.

HÜBSCHER-YOUNGER, T., AND NARAYANAN, N. H. 2002. The influence of authority on convergence in collaborative learning. In *Proceedings of the Computer Support for Collaborative Learning Conference*, Lawrence Erlbaum, 481–489.

HÜBSCHER-YOUNGER, T., AND NARAYANAN, N. H. 2003a. Constructive and collaborative learning of algorithms. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, ACM Press, 6–10.

HÜBSCHER-YOUNGER, T., AND NARAYANAN, N. H. 2003b. Designing for divergence. In *Proceedings of the Computer Support for Collaborative Learning Conference*, in press.

HUNDHAUSEN, C. D. 2002. Integrating algorithm visualization technology into an undergraduate algorithms course: Ethnographic studies of a social constructivist approach. *Computers & Educations 39*, 237–260.

HUNDHAUSEN, C. D., AND DOUGLAS, S. A. 2000. Using visualizations to learn algorithms: Should students construct their own, or view an expert's? In *Proceedings of the IEEE Symposium on Visual Languages*, IEEE Computer Society Press, 21–28.

HUNDHAUSEN, C. D., AND DOUGLAS, S. A. 2002. Low-fidelity algorithm visualization. *Journal of Visual Languages and Computing 13*, 449–470.

HUNDHAUSEN, C. D., DOUGLAS, S. A., AND STASKO, J. T. 2002. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing 13*, 259–190.

LAHTINEN, S-P., SUTINEN, E., AND TARHIO, J. 1998. Automated animation of algorithms with Eliot. *Journal of Visual Languages and Computing 9*, 337–349.

LAVE, J., AND WENGER, E. 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press.

LINN, M. C., AND BURBULES, N. C. 1993. Construction of knowledge and group learning. In K. Tobin (Ed.), *The Practice of Constructivism In Science Education*. Lawrence Erlbaum Associates, 91–119.

NARAYANAN, N. H., AND HEGARTY, M. 2002. Multimedia design for communication of dynamic information. *International Journal of Human-Computer Studies 57*, 4, 279–315.

STASKO, J. T. 1997. Using student-built algorithm animations as learning aids. In *Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education*, ACM Press, 25–29.

TVERSKY, B., MORRISON, J. B., AND BETRANCOURT, M. 2002. Animation: Can it facilitate? *International Journal of Human-Computer Studies 57*, 4, 247–262.

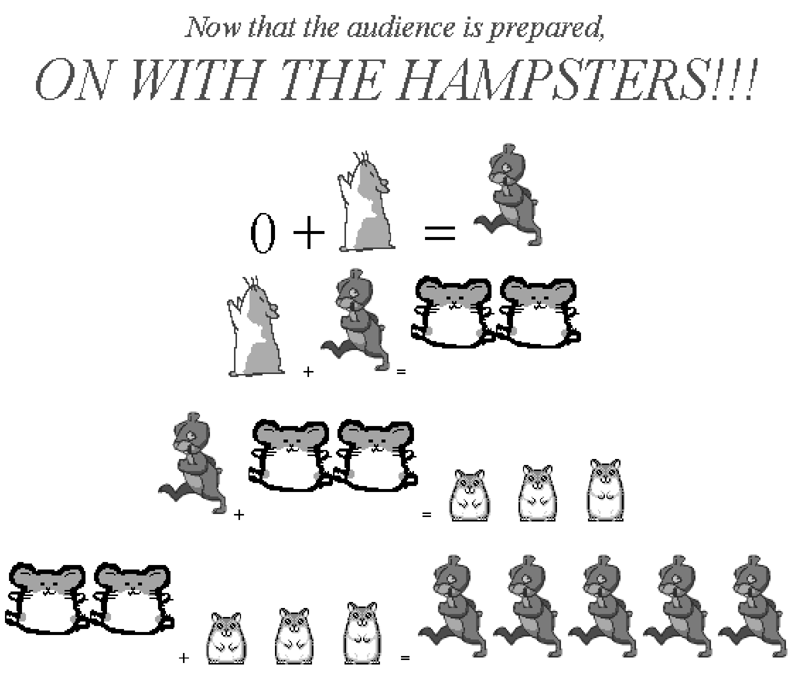
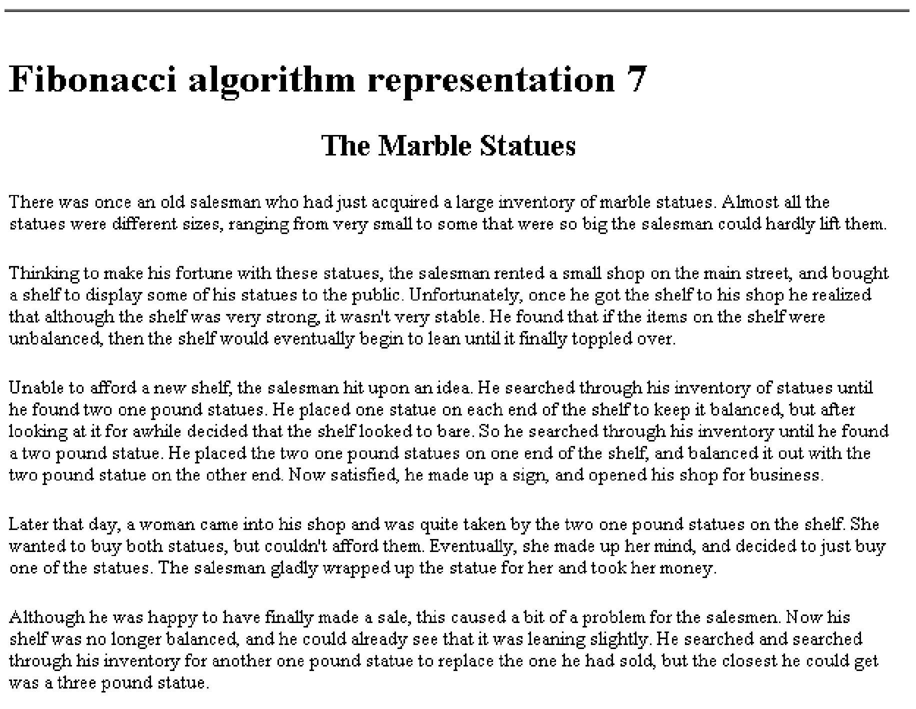


Figure 5. A screenshot from the Dancing Hampsters (sic) visualization of the Fibonacci algorithm.



Fibonacci algorithm representation 7

The Marble Statues

There was once an old salesman who had just acquired a large inventory of marble statues. Almost all the statues were different sizes, ranging from very small to some that were so big the salesman could hardly lift them.

Thinking to make his fortune with these statues, the salesman rented a small shop on the main street, and bought a shelf to display some of his statues to the public. Unfortunately, once he got the shelf to his shop he realized that although the shelf was very strong, it wasn't very stable. He found that if the items on the shelf were unbalanced, then the shelf would eventually begin to lean until it finally toppled over.

Unable to afford a new shelf, the salesman hit upon an idea. He searched through his inventory of statues until he found two one pound statues. He placed one statue on each end of the shelf to keep it balanced, but after looking at it for awhile decided that the shelf looked to bare. So he searched through his inventory until he found a two pound statue. He placed the two one pound statues on one end of the shelf, and balanced it out with the two pound statue on the other end. Now satisfied, he made up a sign, and opened his shop for business.

Later that day, a woman came into his shop and was quite taken by the two one pound statues on the shelf. She wanted to buy both statues, but couldn't afford them. Eventually, she made up her mind, and decided to just buy one of the statues. The salesman gladly wrapped up the statue for her and took her money.

Although he was happy to have finally made a sale, this caused a bit of a problem for the salesmen. Now his shelf was no longer balanced, and he could already see that it was leaning slightly. He searched and searched through his inventory for another one pound statue to replace the one he had sold, but the closest he could get was a three pound statue.

Figure 6. A portion of the Marble Statues visualization of the Fibonacci algorithm.

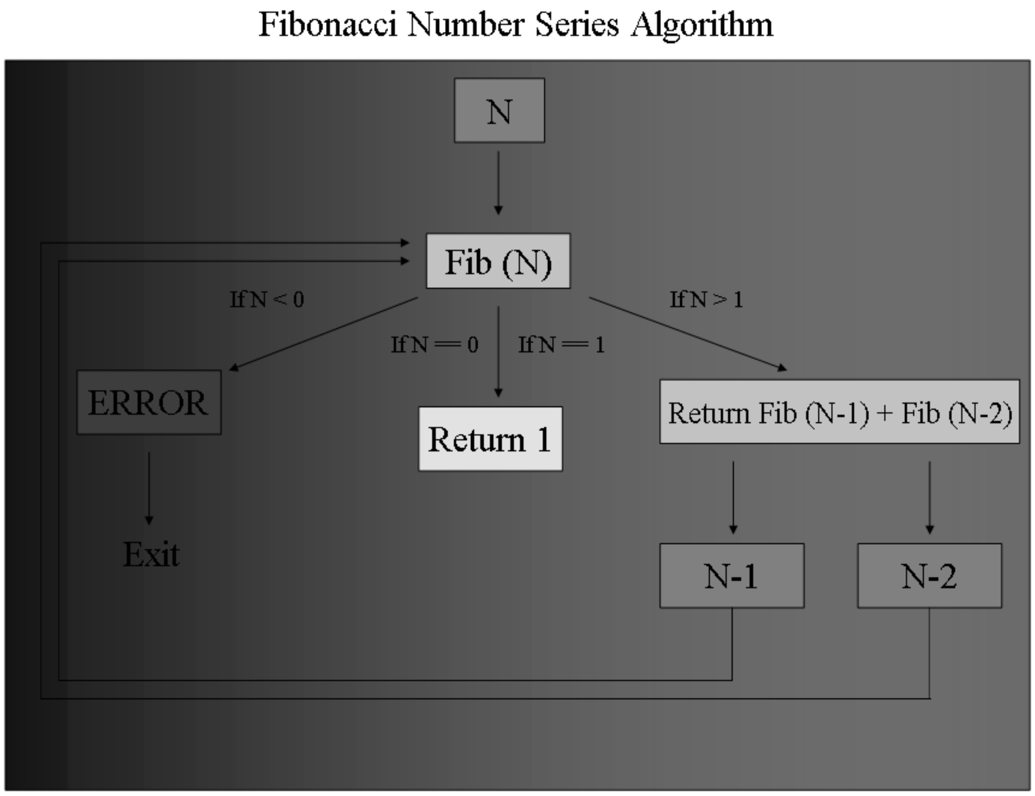


Figure 7. Another visualization of the Fibonacci algorithm.