

LITRE Project Final Report: Integrating Algorithm Visualization into CSC 505

Steffen Heber and Brian Howard

Abstract

This report summarizes our progress so far on the LITRE project “Integrating Algorithm Visualization into CSC 505”. The majority of the implementation goals described in the original proposal have been met. However, part of the final evaluation of the resulting algorithm visualization library will not take place until the spring semester 2008.

1. Introduction

1.1 Description

The goal of this project is to create a library of interactive visualizations for use in teaching CSC 505: Design and Analysis of Algorithms and other related courses such as CSC 316: Data Structures for Computer Scientists. Currently, these classes teach students the use of data structures, fundamental algorithms, algorithm design, and algorithm analysis techniques mostly using conventional educational tools such as black board and overhead projector. Our goal is NOT to replace these tools, but to complement them with dynamic and interactive visual representations of abstract algorithm design and analysis concepts (e.g. growth of functions, recurrences), and animations illustrating how algorithms and data structures work. We believe that providing students with multiple views and teaching techniques will facilitate a better understanding of abstract concepts like the analysis of algorithms. The idea is similar in spirit to Gloor's HyperCard algorithm animation CD-ROM (Gloor et al. 1993) which accompanied the first edition of Corman, Leiserson and Reivest's algorithm textbook. However, instead of HyperCard (Goodmann 1987), which was discontinued in March 2004, our library is written in the highly portable Flash Professional 8 (Macromedia <http://www.macromedia.com/>). This guarantees problem free use virtually independent of operating system and client hardware specifications -- Flash Player is installed on 98% of desktops globally.

1.2 Background

The PI for this project is Dr. Steffen Heber, who teaches and conducts research at NCSU under a joint appointment from the Computer Science Department and the Bioinformatics Research Center. Brian Howard, one of Dr. Heber's graduate students, did the computer programming required for this project.

Dr. Heber teaches a variety of computer science and bioinformatics courses for which the material in this library is relevant. In particular, the visualization library targets CSC 505 Design and Analysis of Algorithms, and CSC 316 Data Structures for

Computer Scientists. The vast majority of students attending these classes are Computer Science and Electrical Engineering students. Both classes usually draw 25-80 students per semester. CSC 316 is primarily for undergraduate students, while CSC 505 is intended for graduate students. Both classes are core degree requirements. Traditionally, these classes are taught in lecture style, with instructors using blackboard, overhead and computer presentation software such as Microsoft PowerPoint, Macintosh Keynote and Adobe Acrobat.

1.3 Objectives

We have three primary goals for this project:

1. To develop a library of interactive visualizations to be used by the instructor to demonstrate and explore computer algorithms and data structures in the classroom during lectures.
2. To make these visualization available for students as a reference outside of the classroom, possibly developing an accompanying set of homework assignments using the library.
3. To determine the effectiveness of this library as an educational tool.

1.4 Student Learning Outcome(s):

Algorithm visualization is widely considered to be one of the most important tools in computer science education. Although it has been shown that the educational effect of simply observing visualizations is unexpectedly low, interactive visualizations which manage to engage students, and which are mixed with other teaching methods like black board or overhead projector, have proven extremely successful (Brusilovsky, and Spring, 2004). It is our belief that the library created for this project has:

1. *Enhanced student comprehension of complex computer science concepts* taught in CSC 505 and CSC 316 by providing additional visual representations (empirical inquiry). Many students do not fully comprehend key concepts of design and analysis of algorithms. We believe that this is often due to the fact that these concepts are communicated on a textual basis, invoking sophisticated mathematical arguments. On the other hand, the learning outcomes of students can often be improved by adding a visual representation; most students are able to grasp information in graphical form better than in textual form (Felder and Silverman, 1988).
2. *Helped students to study and learn independently* about fundamental algorithms (research from sources). Due to time constraints, students learn about many important data structures and algorithms only in reading and homework assignments. Here, in addition to providing a visual representation, visualization has an important motivational character. It can make studying algorithms more

fun by making algorithms less intimidating and more accessible. This results in improvement in students' attitudes towards studying algorithms and leads to better outcomes.

3. *Improved student motivation and participation* in the classroom. By providing a 'change of gears' during an in-class lecture, the use of visualizations has had the effect of recapturing student attention and providing students with an opportunity to interactively participate in the lecture. This appears to improve student motivation to learn the material and increase their interest in the class.

In addition, our library also helped the instructor to efficiently illustrate complex algorithms in class.

2. Method

2.1 Project Design

Currently, our library covers the following subject areas: running time analysis (worst, best, and average case), asymptotic growth classes of functions, recurrences, sorting algorithms (insertion sort, heap sort, merge sort, quick sort, radix sort), graph algorithms (DFS, BFS, Dijkstra shortest path algorithm), and tree search algorithms (in-order, post-order, pre-order). Furthermore, our experience developing and using these custom visualizations in the classroom allowed us to better identify existing visualizations (developed by other groups), that serve to complement the tools that we developed for this project. Examples of visualizations that we were able to identify and utilize in the classroom included: AVL trees, red-black trees, skip lists, and basic data structures such as stacks, queues, linked lists, priority queues, and hashes.

The implementation process for each visualization involves the following steps:

1. *Requirements Analysis and Design Meeting*: During this phase, the course instructor and software developer meet to define the minimum design requirements for the visualization. Generally, this involves a demonstration by the instructor of how the algorithm might be taught in the classroom using a traditional approach (such as whiteboard, PowerPoint slides, or paper models.) Then, after a brainstorming session, the team collaboratively designs a visualization tool (to be implemented in Flash) that will, ideally, enhance this experience from the perspective of both student and instructor.
2. *Prototype Implementation*: After gathering requirements, the developer writes computer code in Flash Action Script to implement the design devised in step one.
3. *Design Refinement*: After the prototype has been developed, the instructor and developer meet again to determine if the visualization produced meets

the design requirements agreed upon in step one. Occasionally, after actually seeing the initial implementation of the design, it will become apparent that further improvements are possible, or that alternate approaches may be more effective. If this is the case, the suggested modifications are then made by the software developer, and reviewed in additional meetings as necessary.

4. *Integration into course materials:* After the software library is complete, these tools need to be integrated into the course teaching material by the instructor. Generally this involves the creation of .HTML web pages that include the Flash visualizations.
5. *Evaluation:* The effectiveness of the individual tools is assessed. This involves measuring the effect on student learning, and ease of use for the instructor.

2.2 Timeline

Summer 2006	- <i>design and implementation of visualizations</i>
Fall semester 2006	- <i>continued design and implementation of visualizations</i>
	- <i>visualizations used in classroom for CSC 316: Data Structures for Computer Scientists</i>
Spring semester 2008	- <i>visualizations to be used in classroom for CSC 505: Design and Analysis of Algorithms</i>
	- <i>final evaluation of visualization library</i>

2.3 Assessment & Analysis Plan

The effectiveness of the visualization library is to be tested in three different ways:

1. *Assessment of students' learning outcomes given additional visual representations.*

There have been many studies related to the effectiveness of algorithm visualizations in class, see (Hundhausen et al. 2002) for an overview. Often students fail to understand what is happening inside a program or an algorithm in the presence of a well-developed non-interactive visualization. It has been shown, that in order to be successful, it is essential to generate interactive visualizations which are integrated with other teaching aids (Hundhausen et al. 2002). To assess the effectiveness of our library in class, we will perform post-teaching tests to measure students' understanding with or without visualization. Our main focus is

to test if the presence of an interactive algorithm animation helps students to learn independently about fundamental algorithms.

We will use a test procedure adapted from (Lawrence et al., 1994). The students will be divided into two equal groups, and both groups will learn about Dijkstra's Shortest Path Algorithm in a lecture. Subsequently, both groups will use the textual description of the algorithm from Corman, Leiserson and Reivest's *Introduction to Algorithms* textbook (Corman et al., 2001). Group 1 will also have access to an interactive animation, while group 2 will get only the diagrams from the textbook. Both groups will complete a multiple-choice test and answer free-response questions designed to articulate concepts relating to understanding the algorithm.

2. *Assessment of students' attitude towards the class.*

The PI will administer an end-of-course questionnaire following instruction of *CSC 316: Data Structures for Computer Scientists*. The questionnaire will elicit student feedback and gauge their impressions of the visualization library. This procedure will be repeated in spring 2008 during instruction of *CSC 505: Design and Analysis of Algorithms*.

3. *Assessment of effort in creating visualizations as well as 'teacher outcomes'.*

Although there is a general belief that algorithm visualizations are helpful educational tools, teachers have been hesitant to use them because they are afraid that they might be too complicated to handle, and that they use up too much class time (Hundhausen et al. 2002). In order to address these concerns, the PI will keep journals with observations concerning:

- Effort generating visualizations and teaching material (lecture and homework assignments) that utilize the library.
- Impact on classroom time and student/teacher interactions when visualizations are used.
- Impact on teaching strategies.

3. Results

At this time, assessment items number 2 and 3 (see section 2.3) have been completed, and our visualizations have been used (and informally tested) in the classroom for the course *CSC 316: Data structures for Computer Scientists*, which the PI taught during the fall semester of 2006. We also tried to perform a controlled experiment to evaluate the algorithms as proposed in the grant (see assessment item number 1, above). We decided that students should be rewarded with bonus points for their

participation in such an evaluation, in order to ensure their best efforts. The most convenient time for the students to participate turned out to be the day after the final exam. Unfortunately, a couple of students were unable to attend our evaluation at this time and formally complained that university policy would prohibit instructors from awarding bonus points after the final. This is actually true, and after we became aware of this fact, student interest in the evaluation collapsed, with only two students agreeing to participate without the prospect of extra credit. Since an evaluation with only two students hardly makes sense, the evaluation was canceled. Our plan is to reschedule the evaluation to occur during an official class period (i.e. before the final exam) in the CSC 505 course which Dr. Heber will teach in Spring 2008. We will then update this report to include the results of that assessment.

The results described below originate from our experiences gained during visualization development, testing during CSC 316, and an end-of-course questionnaire.

3.1 Effect on Student Learning

In our end-of-course questionnaire we first asked about the students' general attitude towards algorithms and CSC 316. Then we posed more specific questions concerning the effectiveness of our visualization project. The results are summarized with respect to the different learning outcomes below; in addition, if available, we also added a collection of 'representative' quotes.

Students' general attitude:

Algorithmic problem solving, correctness proofs, and efficiency analyses are interesting and fun.				
2	10	8	8	1
Strongly agree	Agree	Undecided	Disagree	Strongly disagree

I have a good feeling toward CSC 316.				
5	15	5	4	0
Strongly agree	Agree	Undecided	Disagree	Strongly disagree

Outcome 1. *Enhanced student comprehension of complex computer science concepts:*

Do you feel that algorithm visualization/animation helped you to learn how algorithms work?				
25	2	1	1	
Positive answer	Neutral answer	Negative answer	Did not respond	

Below are some quotes from the questionnaire:

“yes, I am a visual learner”

“yes, because you could try different examples”

“yes, it helped me to visualize better. Also, it broke the monotony of just looking at slides.”

“No, pseudocode is enough for me”

Outcome 2. Helped students to study and learn independently.

Was algorithm visualization for you a useful aid to study and learn independently?			
25	1	1	2
Positive answer	Indifferent	Negative answer	Did not respond

Below are some quotes from the questionnaire:

“yes, to practice how the algorithm works”

“yes, especially reviewing [tree] traversal and [search tree] balancing”

“it is much better to see how the ADTs work instead of just trying to figure out how they work from reading”

“I used them to study for the final”

“probably not, too time consuming”

Outcome 3. Improved student motivation and participation in the classroom.

Did you enjoy algorithm visualization in class?			
25	3	1	0
Positive answer	Indifferent	Negative answer	Did not respond

Below are some quotes from the questionnaire:

“it helped me to understand, and stay awake”

“sometimes, other times they went by slowly, but slow is good for explaining”

“they helped to make the lecture interesting”

“fun stuff!”

3.2 Effect on Faculty/Pedagogy

In general, This project motivated the PI to use many more examples and exercises in the classroom than before. Although preparing the visualizations from scratch takes a very long time, and requires several iterations of design and refinement, using existing libraries actually made preparation more efficient. It is comparable or even faster than preparing a paper or blackboard model. Also, the actual demonstration in class is, in general, simpler, and allows the instructor to spend more time on explanations, or questions. However, one practical problem which occurs occasionally is that the actual manipulation of the visualizations/animations via mouse or keyboard can be awkward and requires the instructor to stand close to the computer.

Using animation and visualization techniques allowed the PI to cover material, like, for example, hash performance over time, or tree update operations, which otherwise would be very hard to teach, and difficult for some students to understand. In addition,

the visualizations also allowed the PI to quickly demonstrate algorithms on multiple levels, e.g. providing a rough overview, or going step by step through the pseudocode.

4. Discussion

4.1 Summary of Important findings

We developed and implemented a library of 17 algorithm visualizations and animations. The visualizations were tested in section 3 of CSC 316 Spring 2006, and the students' attitude towards the visualizations was assessed via an end of class questionnaire. An assessment of the effectiveness of the visualizations with respect to students' learning will be performed in CSC 505, Spring 2008.

In general, the class showed a high level of enthusiasm for design and analysis of algorithms, especially if one takes into account that this material is quite abstract and difficult to understand. The general attitude towards CSC 316, as well as the students' responses on our questions regarding the impact of visualizations on the various student learning outcomes was surprisingly positive. Several students used the visualizations successfully outside class to study on their own, and in preparation for their exams. However, until completion of our evaluation, it remains an open question if this positive attitude is also accompanied by a measurable increase in students' understanding of the material.

Although developing a visualization library from scratch is a time consuming endeavor, the use of visualization in the classroom might even save preparation time once such a library has been created. Furthermore, visualizations provide a good opportunity to 'change gears', and appears to have a positive impact on maintaining students' attention during class. Naturally, some topics, for example hash performance over time, or tree balancing operations, lend themselves to visualization and animation, while other topics such as asymptotic running time analysis might be better taught by conventional methods, but there is hardly any topic which suffers from use of computerized visualization. However, there is a real danger that students may skip over important details without really understanding the algorithm. The PI tried to overcome this problem by promoting class participation at crucial time points, for example by asking questions like "what will happen next?"

4.2 Conclusions

Overall, both the students and the PI really enjoyed the use of visualizations/animations in class. One student remarked: "visualizations are always fun!" The PI has successfully used algorithm visualization to teach CSC 316: Data Structures for Computer Scientists and plans to use the library for an upcoming course, CSC 505: Design and Analysis of Algorithms. Students have used the visualizations to prepare and check their homework assignments, and to prepare for the final exam in CSC 316. In class, our visualizations worked best when they were used as complementary tools involving student activities. Outside the classroom, students preferred

visualizations/animations which were helpful for homework or an exam, intuitive to manipulate, and easy to access. Unfortunately, the final evaluation of the effect on students' learning had to be postponed to Spring 2008, so it is not yet clear if the use of our visualizations significantly improves students' understanding of how algorithms work, or whether the primary effect is only to improve students' attitude towards the material. The PI plans to further expand the use of visualization tools in his teaching, and to continue developing the library. However, it appears that we greatly underestimated the amount of work which is required to develop such a library. There is much room for additional development in this area, and based on the experience gained during this project, the PI plans to prepare a grant proposal to develop a corresponding library of visualizations for Bioinformatics algorithms.

4.3 Assessment of Technical Challenges

- ⊗ We found that there was an initial learning curve as the software developer became familiar with the Flash development environment, Action Script syntax, and the idiosyncrasies of the Flash object model. However, after this initial phase was completed, the pace of development increased.

- ⊗ Unfortunately, Gloor's HyperCard visualization library was less useful as a starting point than we had originally anticipated. Instead, however, we were able to substitute other supplementary materials as references, including the following online materials: <http://ww0.java4.datastructures.net/> , <http://ww3.algorithmdesign.net/> .

4.4 Potential Applications for Others on Campus

In the short term, this project will impact only the student's who take CSC 505 class (enrollment: about 60 students per semester) and CSC 316 Data Structures for Computer Scientists (enrollment: about 170 students per semester). In the long term, however, the suggested algorithm visualizations could also be used in the following classes: CSC 314 Data structures, and CSC 541 Advanced Data Structures (total enrollment: about 45 additional students per semester).

In addition, it would be possible to develop similar visualizations tailored towards algorithms used in Bioinformatics: BI/GS 501 Bioinformatics I, BI 502 Bioinformatics II, CSC 530 Computational Methods for Molecular Biology (total enrollment: about 75 students per semester).

5. References

- Gloor, Dynes, and Lee. *Animated Algorithms*. MIT Press, 1993, CD-ROM.

- Gloor. AACE -- Algorithm Animation for Computer Science Education. In Proc. 1992 IEEE Workshop on Visual Languages, pages 25--31, October 1992.

- Goodman. The Complete HyperCard Handbook. Bantam Books, 1987, ISBN 0966551427.
- Brusilovsky and Spring, (2004). Adaptive, Engaging, and Explanatory Visualization in a C Programming Course. In: L. Cantoni and C. McLoughlin (eds.) Proceedings of ED-MEDIA'2004 - World Conference on Educational Multimedia, Hypermedia and Telecommunications, Lugano, Switzerland, June 21-26, 2004, AACE, pp. 1264-1271.
- Felder and Silverman, (1988). Learning styles and teaching styles in engineering education. Engineering Education, 78 (7), 674-681.
- Hundhausen, Christopher, Douglass, Sarah, and Stasko. A Meta-Study of Algorithm Visualization Effectiveness. Journal of Visual Languages and Computing, Vol. 13, No. 3, June 2002, pp. 259-290.
- Lawrence, Badre, and Stasko. Empirically Evaluating the Use of Animations to Teach Algorithms. Proceedings of the 1994 IEEE Symposium on Visual Languages, St. Louis, MO.
- Cormen, Leiserson, Rivest, and Stein. Introduction to Algorithms (Second Edition) published by MIT Press and McGraw-Hill, 2001.