Building a Community and Establishing Best Practices in Algorithm Visualization through the AlgoViz Wiki

May 19, 2008

(Due May 21.)

Project Summary

Algorithm visualizations (AVs) are widely viewed as having the potential for improving computer science education. But while there exist many documented successes, other researchers have been unable to demonstrate that AVs positively affect learning outcomes. The conclusion is that successful use of AVs depends on both the availability of quality educational materials and their proper deployment. Technically, much has changed since the mid 1990s in that AV implementations are now typically written in Java, and access to computers in the classroom by both instructors and students has greatly improved. Yet even though the community has made steady progress in techniques for developing, evaluating, and deploying AVs, the field has progressed little from the mid 1990s in terms of adoption levels or total educational impact.

Surveys of CS faculty show that impediments to successful use of AVs in the classroom include: difficulties finding worthwhile AVs for use on desired topics; difficulties adapting AVs to a given instructor's classroom setting; and lack of knowledge on the best way to deploy AVs. On the developer's side, our study of a collection of over 350 AVs shows that many existing AVs are of little pedagogical value, and there is poor distribution of topical coverage. This indicates that many developers are not making use of lessons learned.

Since effective ways are known to develop and deploy AVs, these difficulties could be overcome by an effective community of AV developers and users. The goal of our project is to build this community. A steering committee of active members in the field will be established to guide creation of an integrated collection of resources and infrastructure to serve as a communications hub and information clearing house. This will leverage the progress already made by AV developers and users to bring about wider adoption and greater successful impact in the classroom. This community will improve the AV development process by better identifying needs, by sharing information on best development practices, and by providing better access to existing development tools.

We will expand on our preliminary work on the AlgoViz Wiki, `http://algoviz.cs.vt.edu`, to make it into a true community resource. As a direct outcome of this project, the AlgoViz Wiki will become a focus point for locating AVs, releasing new AVs, reading reviews from users, and sharing experiences about classroom use. We will help AV developers and users connect and interact with each other, provide social support for reviewing, rating, and commenting on available AVs, and share activities and plans for how best to use AVs. Resources for developers will include information about development tools and AV toolkits, information about licensing and intellectual property policies, guidance on what works and what does not in AV design, and guidance on what topics are in greatest need of new visualizations. We will develop a SourceForge site where developers can share exemplar source code for AVs, allowing new developers to see how others have solved fundamental design and development problems.

Intellectual Merit: Our efforts will improve the CS education community's understanding of how AVs can be made pedagogically effective. It will leverage the collective knowledge of the community to drive better development practices and better deployment. The community will collectively identify quality instructional materials and effective modes of deployment.

Broader Impact: Improving on the general availability and quality of AVs can affect the educational outcomes for tens of thousands of students every year in computer science and related disciplines. Providing a community resource on best practices will improve the development of future AVs around the world. A clearing house of available AVs and documenting best practices for their use will provide quality educational materials to instructors.

1 Problem Statement

Algorithms and data structures are one cornerstone of an undergraduate computer science education. One potential technique for improving instruction in this critical area is to include algorithm and data structure visualizations and animations (hereafter referred to as "algorithm visualizations" or "AVs") into the curriculum. AVs have a long history in computer science education, dating from the 1981 video "Sorting out Sorting" by Ronald Baeker and the BALSA [4] system from 1984. Since then, hundreds of AVs have been implemented and provided freely to educators, and scores of papers have been written about them [34]. AVs bring algorithms to life by graphically representing their various states and animating the transitions between those states. Good AVs illustrate data structures in natural, abstract ways instead of getting bogged down in memory addresses and function calls.

AVs are naturally attractive to educators, who nearly universally view them positively [20]. They are also consistently "liked" by students. But an important question persists: *Are they effective at educating computer science students?* There has been some debate in the literature as to whether AVs are effective in practice. Some studies have shown the classic dismissal for many technological interventions in education: "no significant difference" [10, 13, 15]. Other studies have shown that AVs can indeed improve understanding of the fundamental data structures and algorithms that are part of a traditional computer science curriculum [17, 5, 11, 9]. Certainly, many AVs exist and are widely (and freely) available via the Internet. Unfortunately, the ones of high quality can be lost among the majority that serve no useful pedagogical purpose (see [30, 8] and data shown in Section 3).

So we see that (a) while many AVs exist, relatively few are of true value, and (b) some AVs can be demonstrated to have pedagogical value, yet it is also quite possible to use them in ways that have no pedagogical effect. These results seem to imply that creating and deploying effective AVs is difficult. There is a growing body of literature that investigates how to create pedagogically useful AVs (for example, [14, 28, 20, 18, 27]). Yet, there is still much to be done before we are at the point where good quality AVs on most topics of interest are widely recognized and adopted.

Instructors report [20] that a major impediment to using AVs in class is the difficulty of finding effective AVs. It is interesting to note that the need for an organized repository for AVs (with some sort of review mechanism) has been recognized since at least 1996 [3]. Yet, there exist no significant repositories of AVs, and certainly none that impose quality control over their content. Within the computer science community, the most likely candidates are the collection of materials submitted to JERIC [16], and the CITIDEL repository [6] (part of NSDL [22]). JERIC and its contributed courseware are indexed as part of the ACM Digital Library [2]. While the ACM DL and CITIDEL are both huge collections, neither appear to have large amounts of courseware in general or AVs in particular. Equally important, neither provide good search tools for courseware or AVs. The bulk of their materials are papers, and they tend to organize by content topic (CITIDEL) or publication venue (ACM DL). Neither supports browsing for courseware separate from the (overwhelming) body of non-courseware content. SIGCSE maintains a collection of courseware links which includes a small number of AVs [31]. Broader courseware repositories include SMETE [32], and Connexions [7], but these are not well known within the CS education community, and they have few, if any, AVs. Further, none of the repositories mentioned here have much "web presence" with respect to whatever AVs they do contain. In all the hours that we have spent conducting Google searches for AVs, not a single AV within any of these repositories was discovered by that means.

Thus, the typical way for instructors to find AVs is to search the Internet. Assuming that a

suitable AV on a specific topic exists somewhere on the Internet, we can hope that a standard Internet search will allow educators to find it. If so, this might alleviate the need to create and maintain specialized repositories or link collections for courseware in general, and AVs in particular. However, as the results from Section 3 indicate, much of what is available is of low quality and is poorly distributed among the topics of interest. Thus, instructors indicate that they are unable able to find a quality AV relevant to what they are looking for [20]. Even if a suitable AV is found, the typical instructor is unaware of what is known about effective ways to use AVs in the classroom.

The bottom line is that AVs have not improved their penetration into CS courses beyond where they were in the mid 1990s. This is despite the fact that the AV research community is making steady progress on understanding how to effectively create and deploy AVs. To quote [18]:

> Previous surveys show a significant disconnect between the intuitive belief that visualization enhances a student's learning and the willingness and ability of instructors to deploy visualization in their classrooms. A key impediment to the adoption of visualizations by instructors is the time required to learn, install, and develop visualizations and then integrate them into a course. Additionally, there is also a perceived lack of effective visualization development tools and software.
>
> Whereas studies have begun to show the conditions under which visualization enhances student learning, the overall educational impact of visualization is and will be minimal until more instructors are induced to integrate visualization techniques in their classes.

We believe that a key reason for this situation is because there exists no community of AV users and developers who can systematically transfer the fruit of research activities into the hands of developers and instructors. Since there is no community, there are no established best practices for developing AVs, no good collections of information about existing AVs and how to use them, and little shared information about where developers can best focus their efforts.

2 Solution Overview

The overarching goal is to build a community of developers and users of AVs who together will improve CS education through the use of AVs. The key to organizing this community will be a collection of online resources that will include (a) the AlgoViz Wiki (`http://algoviz.cs.vt.edu`) that includes user reviews of AVs and an annotated bibliography of the research literature, (b) a SourceForge project that provides open-source exemplars of AV development projects contributed by the community, and (c) community forums and other communications infrastructure. As a direct outcome of this project, CS educators will know where to locating AVs, release new AVs to the community, read reviews from users, and share experiences about classroom use. We propose to accomplish our goals through the following four aims.

We will build a steering committee of active researchers and developers to guide development of the community-support infrastructure. In particular, they will help guide future development for the AlgoViz Wiki and the SourceForge site. Many of the relevant individuals already participate in an informal community through ITiCSE working groups related to AVs [20, 18, 27].

We will extend the core infrastructure needed to support the community: The AlgoViz Wiki, a SourceForge site, and community forums. As described in Section 3, we have

already created the AlgoViz Wiki, the largest collection of links to AVs ever achieved, and the largest bibliography and review collection of AV-related research publications. The Wiki can be leveraged to allow the community to drive improvement through ratings and reviews of AVs. We will create community forums to allow the users and developers to communicate. We will establish a reference SourceForge open-access development site. The SourceForge site will include both open-source example AVs and development tools contributed by the community (see Section 4.3).

We will develop and support a community of users who employ AVs in their teaching. We will help AV users connect and interact, rate and review available AVs, and share activities and plans for how best to use AVs. By providing discussion forums and a mechanism whereby users can rate and review AVs, we hope to establish an active community that will drive the field forward.

We will build a community of developers of AVs. While many good AVs are available, the need for more and higher quality AVs continues. As explained in Section 4, current AV developers typically do not make use of good software engineering practices or development tools. They clearly do not know the topical coverage for existing AVs, since they keep repeating the same topics. They tend to make the same mistakes in AV design. There currently exist no community-wide online forums or other mechanisms for exchanging information among AV developers. Many problems can be mitigated by the existence of an easily accessible website that provides AV users and developers with a way to communicate, and that carries the "corporate knowledge" of the user and developer community.

3 Preliminary Results

3.1 The Algorithm Visualization Catalog

Since Spring 2006, we have made a significant effort to catalog as many existing AVs as we could on topics related to data structures and algorithms. The results can be found at our Data Structure and Algorithm Visualization Wiki [35] (the AlgoViz Wiki itself is further described in Section 3.2). We have developed the most extensive collection of links to AVs currently available, with nearly 400 links as of May, 2008. While it is by no means complete, this collection can serve as a representative sample of the total population of AVs accessible on-line. A number of interesting research questions can be addressed by analyzing the contents of the catalog. In this section we present some of our initial findings [30, 8]. Many of these statistics come from a careful analysis of 127 sorting AVs, which we believe is a reasonable sampling of the entire data set.

How did we find them? We began with a list of all AV systems that we were aware of from our knowledge of the field. We developed a topic list based on our experiences teaching relevant courses. We considered what search terms would be most productive for locating AVs via Internet searches. We then performed searches using Google to find whatever we could. We examined the pages we found to try to locate other AVs, since developers of a given AV often have others available. Sometimes we could find other AVs from direct references on the pages we had, and other times we could deconstruct URLs to find more AVs. Whenever we came across a page that had links to collections of AVs, we would include any new ones not yet in our collection. We speculate that we have so far captured at least half of what is publicly available, and have plausibly located the vast majority of easily found, better-known AVs. The search procedure models the process a diligent educator might employ for locating an AV; if an AV cannot be found by our method,

it seems unlikely the typical instructor would locate it via standard Internet search engines. Our team is still actively collecting new links.

How many are available? As of May 5, 2008, the collection contained links to 382 AVs. Many of these are individual applets or programs, but a significant fraction appear as parts of integrated visualization collections (typically, individual applets that embody 5–20 distinct AVs, or toolkits that distribute a collection of 5–20 distinct AVs as a unit). If a given applet or program contains multiple AVs (for example, a single Java applet that embodies separate AVs for both stacks and queues), it is counted multiple times—once for each distinct visualization. There are well over 200 distinct programs or applets. These totals do not include older systems like XTANGO and Balsa. We have not yet integrated their AVs into the main catalog, although the Wiki does contain pointers to these systems. Including these older AVs would bring the catalog total to around 500.

How are they implemented? Virtually all AVs and toolkits implemented since the mid 1990s have been implemented in Java. Nearly two thirds of AVs appear as applets directly embedded in web pages. Nearly one third are Java applications that must be downloaded and opened locally. Only about 5% are implemented some other way. These numbers are somewhat biased. There is a tendency for us to search for applets, since these are easier to find (including "applet" as a keyword often results in locating additional AVs on Google searches). AVs distributed as applets—i.e., that can be embedded directly in web pages—typically get more attention from potential users, since they need not go through the additional step of downloading and unpacking. We do a significant amount of cross checking by capturing links from AV link collections that we find to insure that our data collection process does not focus unfairly on individual applets. However, our catalog is currently known to be deficient in that it does not yet include pre-Java AVs.

How are they disseminated? We found almost no AVs in large, organized courseware repositories. Many AVs are cataloged by link sites, meaning sites that (like our Wiki) attempt to link to collections of AVs that the site managers have considered worthy. Most of these link sites are small, perhaps linking to 20 favorite AVs for some course or textbook. There have been other efforts to produce comprehensive catalogs of AVs (the Hope College collection [12] and Guido Rößling's animation repository [1] containing approximately 230 AVs).

Who makes them? A little over a quarter of AVs are essentially single efforts by their respective authors. Another fifth are provided by "small shops" that have created 5–15 AVs, often as individual Java applets. These might have each been created by the same individual over some number of years (typically a faculty member who is teaching relevant courses), or they might have been developed by a small number of students working for a faculty member. Over half of the AVs in our catalog are created by groups who have 15 or more visualizations under their belts. Most such AVs are part of a system, but a significant minority are parts of collections we cannot characterize as a system. Note, though, that once a single AV from a collection or small shop was added to the Wiki, the entirety of that collection would typically be discovered and added. This ensures that we probably have AVs available from groups with at least a little visibility. The AVs we missed are presumably heavily skewed towards one-offs.

What is the content distribution? Since we have limited resources, we have so far restricted our study to topics typically covered in undergraduate courses on data structures and algorithms. While this concerns mostly lower division material, we also included some upper division topics like computational geometry, $\mathcal{NP}$-complete problems, dynamic programming, and string matching. A histogram of our top-level categories for grouping AVs is shown in Figure 1. We see that there is

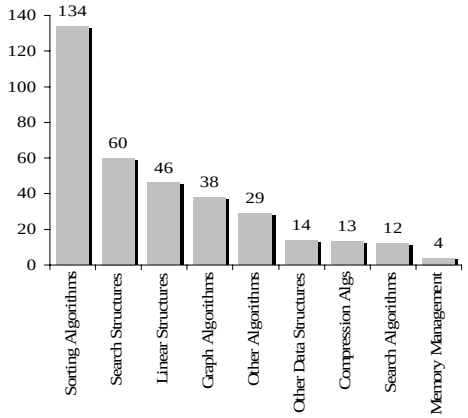

Figure 1: Histogram of major categories for AVs in the Wiki catalog.

wide variation in coverage. Over 10% of all AVs that we have found are on linear structures such as stacks and queues, even though these probably present less difficulty to students than many other topics. Over a third of all AVs are on sorting algorithms. Sorting is an important topic in the undergraduate curriculum, but it is certainly being given too much attention by creators of AVs. Further, many of the sorting AVs are variations on the classic "Sorting out Sorting" video, and just show bars being swapped. In contrast, most specialized and advanced topics are poorly covered. There is certainly room for new AVs.

What is their quality? We made a careful analysis of the 127 sorting AVs in the catalog at the time. Only 15% achieve a rating of "recommended" for use, either as a lecture aid, as the basis of a lab exercise, or for self-study of a topic. Another 45% are rated as "has potential" but are severely limited in pedagogical or functional ways. The remaining 40% receive a rating of "not recommended." They fail to function at all or have little redeeming value for educators. Even the better AVs tend to have serious deficiencies. Perhaps half of all AVs are actually animations—essentially, real-time-rendered movies with little or no meaningful interaction. While some animations are useful, in general, users of animations are relegated to being passive observers with no control over pacing (aside from animation speed), the data being processed, or the operations being conducted. A different type of deficiency often occurs with AVs of tree structures. Typically, these will show the tree that results from a user-selected insert or delete operation. Rarely do they illustrate at all, let alone effectively, how the insert or delete operation actually works. It is exceedingly rare that any AV goes beyond explaining how something works, to explaining implementation pros and cons or performance characteristics.

When were they made? Some well-known systems for creating AVs were developed in the early 1990s. However, most of these are no longer available or so difficult to access due to changes in operating systems that they are not currently a factor in education. Considering the development of AVs since the mid-1990s (i.e., those in Java), there is a decline over time in the creation of new AVs, particularly after 2002. Figure 2 shows a histogram of the last-change dates for the sorting AVs currently in the collection. These counts are "by project" rather than by individual AV, in that if a given AV or AV system provided visualizations for multiple sorting algorithms, then it only counts once in the histogram. While there are still active projects, overall activity does not appear to be as extensive today as it had been previously. Since the catalog records the "last updated" date for the various AVs, the better numbers at the end indicate that there still exist some active

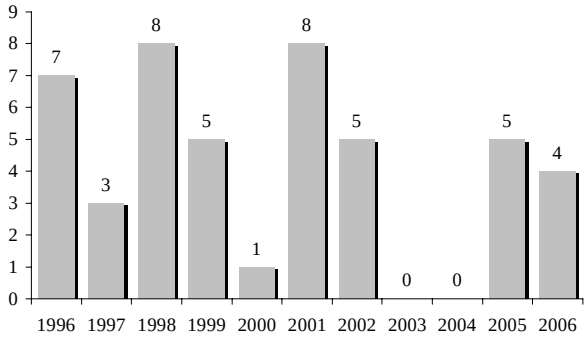

Figure 2: Last-change dates for sorting visualizations. Counts are by project.

projects, as opposed to a recent rise in activity. The recent decline in the creation of new AVs certainly cannot be explained by saturation of either topic coverage or content quality, since both are sadly lacking. Ongoing projects appear to be no more attuned to gaps in topical coverage. We speculate that students were readily available to create AVs during the Java boom of the mid to late 1990s, when Java and applets were new and the "in thing" for students to learn. But now there are other "in things" competing for students' attention. The nationwide drop in computer science enrollment might also mean that there are fewer students available to do such projects. This is not necessarily a bad outcome, as such student-driven "one-off" implementations tend to be of low pedagogical value. It could be that the larger projects are ongoing at similar rates to 10–15 years ago. More sophisticated data collection and better catalog entries that will result from this project can answer such questions.

Will we find them again? Like everything on the Internet maintained by individuals, AVs have a turnover rate in their accessibility. To measure of this, we track the status of the links on our Wiki. Each week, a snapshot of each link's accessibility is recorded. At one point early in the history of the catalog, there were 46 working links. After three months, one of those 46 had a host machine that was no longer on the Internet, three returned "page not found" (two of these three were at the same site), and two became permanent redirection pages (indicating that the links in the Wiki need to be updated or eventually those AVs will be lost). Thus, over a span of three months, 6 of 46 links (13%) were either lost or moved. On July 31, 2006 the catalog had 251 links. By February 26, 40 of those had disappeared and 13 became redirects. Thus, for this seven month period, 21% were lost or moved. On November 27, 2006, the catalog held pointers to 382 links, 23 of which were not working. By February 26, 2007 (three months later), 84 were lost and 4 became permanent redirects. For this period, 23% of the links were lost or moved, while 3% reappeared. These numbers expose an extremely high turnover rate, and presents a major challenge to creating a useful index of AVs. Fortunately, some of the lost links were recoverable with some searching—they had been moved and not redirected from their old links. Our most current run (May 5, 2008) showed that of 382 entries in the catalog, 316 were found, 50 were redirected, and only 16 were lost (a loss rate of just over 4% over the past year). These better results are in part due to our efforts to chase down and restore bad links when possible.

3.2 The AlgoViz Wiki

A Wiki is a user-editable website, usually supporting a simplified markup syntax and directed towards a particular topic. The AlgoViz Wiki is implemented using the MoinMoin Wiki Engine [36].

MoinMoin was selected because Dr. Edwards (Co-PI on this project) had already evaluated several Wiki packages and settled on MoinMoin for his other work. He administers the AlgoViz Wiki on his existing Wiki "farm." The Wiki concept is preferable to a standard HTTP server for this project because it allows for the community access and editing we desire (see Section 4 for more details).

Analysis Tools We have created various tools to automatically extract data from the AlgoViz Wiki, including a database query tool, a weekly link checker, and a counter for updating the top-level catalog counts. The query tool [26] loads each visualization entry and reformats its data as a row in a CSV (comma-separated values) dataset. The dataset is timestamped and archived. When the user selects a dataset and a column label, the tool returns a histogram of the visualization entries by that column. Our weekly link checker searches the Wiki's catalog, recording the HTTP status code for each link. This allows us to track AV disappearances over time. The checker's output (page name, URL, and status code) is stored as an XML file with a time and date stamp and weekly runs are archived to create a historical record.

Other cataloged software In addition to AVs, the Wiki contains a catalog of toolkits for authoring AVs and a small collection of visual debuggers. Many of the toolkits are also collections of AVs, so these are represented multiple times: once for each AV they contain and once on the page describing available toolkits.

Bibliography Another major component of the Wiki is an annotated bibliography containing over 80 research papers about AVs. One of the hardest things to do in any field of study is to acquire the knowledge and the zeitgeist of the field's community. We hope that the bibliography will provide a starting point for newcomers to the field and a repository of important work to which practitioners can refer. Many of the papers listed have summaries, and nearly all have BibTeX citations for easy referencing. A topical index helps to locate a particular paper.

Current Wiki traffic The AlgoViz Wiki is still being developed and so has not been heavily advertised, but we have been collecting web server traffic statistics. After factoring out search engine spiders and other chaff as best we can, our data indicate that the Wiki serves approximately 30-40 page requests per day to Non-Virginia Tech users, with a steady rise in accesses over the past four months. As a result, the Wiki has already established a growing degree of external visibility. Our greatest source of referring traffic comes from Wikipedia, where a few articles on data structures refer to the Wiki. This suggests that one strategy for increasing visibility will be to increase the number of relevant articles at Wikipedia that reference the AlgoViz Wiki. Our second largest referral site is Google.

4 Developing a Community for Algorithm Visualization

4.1 A Steering Committee for AV Community Development

A major step toward developing an AV community will be to initiate a "steering committee" to guide policies and the technical foundations of the project. This committee will be drawn from people who have been active in the AV field, including AV developers, researchers, and participants in ITiCSE working groups related to AVs [20, 18, 27]. To initiate the steering committee's efforts, we plan to hold a one-day workshop of major players in the AV development and user community. Funds for this workshop are being requested from NSF under a separate proposal. We plan to hold this meeting in conjunction with SIGCSE 2009 in Chattanooga, TN. We will use this meeting to

brainstorm ideas on community building, organization and management of the Wiki, and building support for the SourceForge site. Hopefully, most workshop participants will agree to become members of the initial steering committee and/or editorial board for the Wiki. Of course, the exact structure for involving more people into the Wiki management—i.e., whether an "editorial board" or some other structure—would itself be a major topic of discussion at the workshop. We will also discuss other ideas such as establishing awards to get people involved in building better AVs and publicizing successes. Awards could also be given for use of AVs in classes or evaluation, or publishing research papers.

4.2 Community Support for Users

We seek to build a community of AV users. Our choice of a Wiki (as opposed to a traditional website) was driven in part by this goal. Obviously users are supported by the content of the catalog, the bibliography, and the other material related to best practice in the development and use of AVs. But we also seek community involvement in assessing and improving the state of the practice. One mechanism for this is the rating system for AVs in the catalog. Quality assessment information in the catalog comes in three ways. 1) An overall ranking (recommended, has potential, not recommended) for each AV. This is given by the Wiki "editors." 2) Ratings by users on a five-star basis, similar to product ratings at a site such as Amazon. 3) Description and review information for the AV. While currently these come from the Wiki editors, in future we hope the bulk of this material will come from the community (perhaps mediated by the Wiki editors) through users submitting reviews and comments directly to the Wiki. Through reviews and ratings, we hope that the the community will eventually define its own opinions on AVs, thus driving developers to provide more effective materials.

A second mechanism for supporting the community is to establish a rich set of discussion forums. Forums would allow instructors and developers to engage in public, threaded exchanges of information and views. If forum activity reaches critical mass, it could become a focal point for establishing a real sense of community presence.

Another mechanism for support is the additional information provided at the Wiki. For those who wish to learn more about the field of AV research, the annotated bibliography should be an important reference center. We plan to include summaries and reviews of all listed publications. We already have a review form mechanism in place that allows the community to contribute to this resource.

Information on state of the practice should become important to developers. As discussed in Section 4.4, typical developers may well be students doing a semester project or faculty doing a small number of AVs essentially on their own. As such they probably are not well versed in mechanics such as open-source licensing, sites to support keeping their projects alive, and version control. All of this information can be provided by the Wiki. Likewise, the Wiki can provide "how to" information on best pedagogical practices, based on synthesizing the existing research on the matter.

The Wiki has proved to be a useful resource for educators teaching courses on AV design and implementation. Already two instructors (Cliff Shaffer and Tom Naps) have taught graduate seminar classes where the students provided reviews and other updates to the Wiki as part of their classwork. We will encourage other instructors to make use of the resource and accommodate their students by providing Wiki edit permissions.

4.3 The Algorithm Visualization SourceForge Site

An important tool for AV developers is good examples of existing AVs, including source code. As part of this project, we will develop a site at SourceForge.net. This SourceForge site will contain a collection of AV implementations. They will be completely open source, and will be contributed by existing developers in the community. The Virginia Tech group will seed the site with source code for all of our Java-based AVs. Tom Naps has agreed to contribute source code from his Jhavé and GAIGS projects (see his letter of support with this proposal) [19, 21]. We will solicit many more contributors. The site would have at least two purposes: 1) Provide reference implementations for good quality AVs that others can use to see how things should be done. 2) Provide a hosting site for AV contributors. The site itself would serve as an important example to developers how to set up their own site and manage their own project. Developers might choose to make use of our site as an "incubator" since we would provide all the necessary infrastructure (via immediate use of our existing SourceForge site with version control set up, a licensing policy in place, etc.) before starting their own SourceForge project. Potentially, our SourceForge site could become a major repository of AV source code in its own right.

Another feature of the SourceForge project site will be an organized collection of solutions to specific technical implementation issues that are regularly encountered by AV developers. By providing both source code and written solution descriptions, AV developers can avoid "reinventing the wheel" in many cases, thus greatly speeding development. In our own AV development projects, we often wished that there were a ready source of expertise on how to address various technical issues. Examples include: tree and graph layout; animation base classes for moving things around on the screen (text, basic graphical objects like circles and boxes); state transition management and the relationship between algorithm states and the AV user interface controls; and creating "help" and documentation screens.

4.4 Best Practices for AV Developers

Our examination of the software development practices employed by AV developers [8] indicate that a great deal can be done to improve the state of the practice in AV implementation. The key is providing relevant information to developers. The AlgoViz Wiki has the potential for providing many resources that could prove critical to efforts to improve AV design and implementation.

Open-source Licensing We have conducted a study of the source code licensing practices for the sorting AVs in our collection [8]. Of 125 AVs in the collection at the time, 43 had source code available (or potentially available). Only 15 were released under Open Source Initiative-approved licenses [25] or something similar. No AVs were found that were explicitly released as public domain, though 18 of the AVs post their source code with no license or other explanation of their redistribution conditions. Many source-available AVs seem to lack a license not because the developers wish to prevent redistribution, but due to inadequate understanding of the process. It makes a practical difference whether or not AV developers think to post a license declaration along with their source code, since any changes made to the source of an AV without an open-source license cannot be redistributed without potentially violating copyright.

Unfortunately, understanding and choosing between the myriad competing open-source licenses available today can be a daunting prospect for a given faculty member or student who wants to post their AV and its source code. Further, many universities and companies have conflicting policies about how institution-funded intellectual property can be distributed. We envision a community

resource in the form of Wiki pages that explain the major differences between the various open source licenses, including rights and responsibilities, written in multiple formats to appeal to different stakeholders. The hope is that with a properly reviewed resource, AV developers will be better equipped to address licensing and IP issues.

FOSS tools and best practices Since AVs are typically given away free on-line along with their source code, they would seem to be a natural target for use of development tools and practices typically associated with the free and open source software (FOSS) community. Moreover, the FOSS community might serve as a model for the AV developer community to emulate. Developers of AVs might take advantage of the large number of tools and best practices available from the open source world if they were better aware of what was available. The Wiki could provide rationale on why to use such tools, and information on how to do so.

An example is source code version control tools such as the Concurrent Versions System (CVS) and Subversion (SVN). Version control allows software developers to track changes to bodies of source code, connect those changes to individuals, and collaborate across physical boundaries (distance, time zones) using the Internet. CVS and SVN are both easy to set up. This small investment of time opens up development to other people at different institutions, thus potentially spreading the cost and effort across a larger number of players. Also, when a build of the code breaks, developers can go back to the last good build and figure out what changes were checked in and by whom. Finally, this source code repository can be used either for open or non-open projects, as CVS and SVN both have built-in authentication to control who can commit changes or browse the code.

As our data indicate (see Section 3), a major problem is the lack of persistence for AVs maintained by individuals at their own sites. Even though universities are typically willing to host faculty or student projects, once the developers leave that university, their site is essentially lost. For projects needing a permanent home on the Internet, there are two widely-known choices: SourceForge [33] and GNU Savannah [29]. SourceForge provides web space, a compiler farm, version control tools, mailing lists, bug tracking, and other infrastructure to ease software development. Savannah provides many of the same facilities. Both require hosted software to be developed under an open source license. By presenting the options and explaining the benefits of permanent Internet hosting, we can hope to cut down on the level of instability associated with individual hosting of websites.

Community pressure and support With good resources and clear explanations for best practices available at the AlgoViz Wiki, and with an established, active AV community, we hope that the very existence of that community would pressure new AV developers to adopt such best practices. Community pressure would get some AV developers over the hurdle of learning about licensing, source version control, and other best practices. Community support would provide them with the tools they need to jump those hurdles. A little education for developers would go a long way towards easing the difficulty of AV project management. Finally, the open-source community itself provides educational benefits to the students who are doing the software development on AV projects. It helps them learn to work in a distributed team, building a "real" codebase [24], and facing the same issues they will face as professional software developers.

AV Toolkit support There exist many toolkits and libraries that could be useful for AV developers. Using an existing AV toolkit could help developers speed their development efforts. The AlgoViz Wiki already includes a rudimentary catalog of available toolkits. By expanding this catalog and documenting the abilities of the various toolkits, there is a potential for significant increase

in use of such toolkits, and a lowering of the bar for developers. By making the mechanics of AV development easier, perhaps developers can then devote more attention to the difficult aspects of designing and producing *pedagogically effective* AVs.

AV pedagogical effectiveness We will provide to developers a practical introduction to the research results on what does and does not work in AV design. This material will be culled from our experience and that of the steering committee, as well as the research literature (for example, [14, 28]). In this way, we can hope to improve the design of AVs, whereas the other resources described above apply mainly to improving the implementation of AVs.

5 Dissemination

The impact of this proposal goes beyond the typical dissemination of results via publications in conference/journals and workshop or tutorial presentations. As a web site, the Wiki itself is all about dissemination of information.

Our first major step toward dissemination will be to build the steering committee. Our efforts on this point are described in Section 4.1, and include a one-day meeting near the beginning of the project (in conjunction with SIGCSE 2009). We will continue to build community through birds-of-a-feather sessions at SIGCSE and relevant working group proposals at ITiCSE.

We will connect the Wiki to the National Science Digital Library (NSDL) through CITIDEL. We have agreements with the directors of the CITIDEL project to index the AlgoViz Wiki catalog within CITIDEL. (A support letter from Boots Cassel, a director for CITIDEL, is included with this proposal.) We will modify the Wiki to support the Open Archives Initiative protocol for metadata harvesting [23] so that CITIDEL can automatically collect data from the catalog.

Dissemination of results and knowledge from this work will come in the form of publications and workshops/tutorials at National conferences. We expect to continue publishing the results of this research at SIGCSE, ITiCSE, FIE, CCSC and/or ASEE, all recognized conferences that cover advances in computer science education. Relevant journals include ACM's *Journal of Educational Resources in Computing* (JERIC), Elsevier's *Computers & Education: An International Journal*, IEEE's *Transactions in Education*, and Taylor & Francis' *Computer Science Education*. As we further develop "best practices" for AVs, it would be natural to give a tutorial on AV design and implementation at (for instance) the annual SIGCSE conference. We are well positioned to do so, as Dr. Shaffer has developed relevant materials for a graduate-level advanced topics course that he taught on AVs in Spring 2008. We held a birds-of-a-feather session related to development and use of AVs at SIGCSE 2008. This was well attended, and we intend to repeat this in 2009 and 2010. It would be appropriate to write a book on AVs for the broader community based on the results of this project. We will also engage the SIGCSE listserv—not just advertising the Wiki there (which we have done), but also initiating and supporting discussions on various aspects of AVs.

6 Evaluation

The project has four principal objectives:

1. Develop and support a **community of users** who employ AVs in their teaching.
2. Develop and support a **community of AV developers**.

3. Provide a **central catalog** of available AVs and an **annotated bibliography** of the AV-related research literature.
4. Develop the **SourceForge project site**.

Figure 3 shows how we will measure the project objectives for this work. We will evaluate objectives regularly throughout the life of the grant, to insure that our project remains on track. The following is a list of the assessment instruments to be used, and goals that we hope to achieve.

- **Establish the steering committee:** Success can be measured by number of participants and by their level of activity. Goal: 10-20 members of the steering committee, with at least half being active contributors.
- **AlgoViz Wiki Site traffic:** Tracking the total page hits, number of referring sites, and other access statistics will provide a measure of site use. By subdividing access metrics based on the section of the site being used—the AV catalog, AV reviews, bibliography, developer forums, and pages on our own AV development work—it will be possible to gauge participation levels for both educators using AVs and developers working on new AVs. Goal: A 10-fold increase in site traffic over present rates.
- **Ratings and reviews:** In addition to basic Wiki traffic, we will also track the number of ratings and user reviews provided, together with the number of unique user accounts that participate in ratings and reviews. Goals: Complete all catalog entries with "editor" ratings and descriptions. User ratings for at least half the catalog entries.
- **Outbound links:** To measure the degree to which educators are using the Wiki to find AVs for their own courses, we will also measure the number of click-throughs on outbound links—that is, the number of times links directed to each external AV site are clicked by users. In addition to providing insight into community impact, this measure will also provide useful guidance on which AVs are most popular and are receiving the most attention. Goal: Evidence that at least a quarter of cataloged AVs have been accessed.
- **Inbound links:** In addition to outgoing links, we will also track and measure the number of referring sites–that is, sites out on the Web that link directly to the AlgoViz Wiki. Goal: A 10-fold increase in inbound links over current levels.
- **Registration counts:** We will measure and track user demographics for the Wiki via registration information that participants create. Goal: At least 100 contributors.

	Objectives			
Measures	Catalog	Community of Users	Community of Developers	SourceForge Site
Site traffic	X	X	X	X
Review counts	X	X		
Outbound link accesses	X			
Inbound link counts	X	X		X
Registration counts		X	X	X
Forum traffic		X	X	X
New AV creation	X		X	X
Reported AV use	X	X		X

Figure 3: Mapping of evaluation measures to defined project objectives.

- **Forums:** In addition to basic registration information, we will also track the number of registered users who participate in the various user forums, both for reading and posting. Goal: At least 100 contributors.
- **New AVs created:** Throughout the project, as new AVs are discovered and cataloged, we will also track whether or not their creators are registered with the AlgoViz Wiki. We will also track the new AV link contributions that our users provide. Over time, these measures will provide a clear picture of how strongly connected the developers of new AVs are with the AlgoViz Wiki and its resources. Goal: Acquire clear baselines for current activity levels, and show a clear upward trend in activity after two years.
- **Reported AV use:** We will continually collect information about the level of use of AVs in Computer Science courses. We will collect information from the SIGCSE email list, the SIGCSE annual conferences, and our own forums and contacts. Goal: Acquire clear baselines for current levels of use, and show a clear upward trend in use after two years.
- **SourceForge site:** We will be collecting software contributions. Natural measures are the number of AVs and contributors. Goals will be set by the steering committee.

7 GRA Support

The major budget request for this project is graduate research assistantship support for a total of 21 months. This resource will be allocated to the following tasks at the indicated level of effort in terms of total months worked over the life of the project. Note that a timeline is not useful here as many of these activities are intermittent and interleaved.

- Support preparations and implementation for the steering committee meeting. 1 month.
- Establish and maintain the SourceForge site and the forum site. 3 months.
- Extend the AV catalog. This includes the substantial effort of completing the details for all of the current entries, incorporate AVs from pre-Java toolkits that remain available, and broaden the coverage beyond AVs related to data structures and algorithms. 4 months.
- Expand and maintain the collection of online tools for managing and supplementing the Wiki. This includes tools for examining the AV catalog database (generating graphs and other analyses). 2 months.
- Produce measures of site traffic (for the Wiki, the SourceForge site, and the forums) and conduct other data collection as described in Section 6 for the purpose of project evaluation. 4 months.
- Provide and maintain all the mechanisms necessary to support community-based rating and reviewing for the AV catalog and the bibliography. 2 months.
- Expand on the bibliography, and include review/summary material for all entries. 1 month.
- Develop and maintain guides for developers. One will relate to mechanics and software engineering-related best practice topics (open source licensing, version control, use of persistent development sites such as SourceForge). Another will discuss best practices for AV design and content based on the research and experience of ourselves and others. This will include a synthesis of published literature on the topic. 3 months.
- Other efforts related to advertising the Wiki and SourceForge sites to potential users. 1 month.

8 Broadening Participation: Minorities and Undergraduates

This project is a natural for recruiting undergraduate students for research projects. We have had many students work on AV-related activities over the years. For example, we had two independent study students during Spring 2007, three interns during Summer 2007, and two more independent study students during AY 2007/8. This was in addition to several graduate-level independent study students. At Virginia Tech, we have had success in broadening participation in computing research by minorities. The PIs have a strong track record working with female and minority students at VT (including two of the four undergraduates doing independent study in the past two years on AV-related projects and one minority summer intern). The Department of Computer Science holds an annual research symposium for undergraduate research students. This is part of the Virginia Tech Undergraduate Research in Computer Science (VTURCS) program. We (the PIs of this grant) often require that undergraduate students that work with us participate in VTURCS with a poster presentation. Dr. Edwards is a co-director of VTURCS.

The CS department generally has had success recruiting African-American female students from neighboring Historically Black College and Universities (HBCU). We have two recent female African American PhD's in computing, both of whom are now in academic positions (Dr. Tracy Lewis, at Radford University, and Dr. Cheryl Seals at Auburn University). We currently have another female African American PhD candidate, Jamika Burge, who recently received a prestigious IBM PhD Fellowship. Another current female African American PhD candidate in HCI, Kayenda Johnson, received a Ford Dissertation Fellowship for Diversity; she is 1 of only 35 students nationwide awarded this fellowship. We currently are recruiting African-American students from Auburn University, as part of our collaboration with Dr. Juan Gilbert. We have received applications from several African-American students for next year's incoming graduate student class. The success of women and minorities in Computer Science at Virginia Tech are detailed at `http://www.womenandminorities.cs.vt.edu/`.

We will also request an REU supplement to increase undergraduate student involvement from outside Virginia Tech. Our existing connections with a number of HBCUs (including Norfolk State University, Virginia State University, and Auburn University) can be used to recruit undergraduate REU students. Our existing REU program in Human-Computer Interaction can serve as a model for our efforts.

9 Key Staff

Lead PI Cliff Shaffer, Professor of Computer Science at Virginia Tech, has interests in algorithms and data structures, visualization, problem-solving environments, and educational use of computers. He was the PI and lead designer for the highly successful **Project GeoSim** in the mid-1990s, which developed a series of simulations for geography education. He has extensive experience with designing problem solving environments for a number of scientific and engineering applications. He is currently a member of the CS departmental undergraduate program committee, and was chairman of the graduate program for eight years. He is the designer and course coordinator for CS2606: Data Structures and File Processing, and wrote the textbook used in that course. Role: Dr. Shaffer will provide overall project management, lead the efforts to create AV developers' guides, establish and interact with the steering committee, and oversee day-to-day management of the Wiki and SourceForge site development efforts.

Co-PI Stephen Edwards, Associate Professor of Computer Science at Virginia Tech, has interests in component-based software, automated software testing, and educational uses of computers. He is the lead designer of Web-CAT, an extensible, automated online program testing and grading system. Dr. Edwards is responsible for Web-CAT's adoption at other institutions within the academic community. He is also a member of his department's undergraduate program committee, and chair of the subcommittee on curriculum and courses. He is designer and course coordinator for CS 1705: Introduction to Object-oriented Development I (a CS1-level course using Java, objectdraw, and student testing activities). Role: Dr. Edwards will manage the technical aspects of the Wiki system, and lead the online community development aspects.

Graduate, undergraduate, and intern students This effort has historically had great success in recruiting undergraduate and graduate students to contribute. We regularly sponsor undergraduate research projects to develop AVs. The initial Wiki was developed by Matt Cooper for his MS thesis. We had two interns from India working on AV development during Summer 2007 and another coming in Summer 2008. Five independent study projects were completed over the past two years on AV-related topics, and another will be completed in Summer 2008. With CCLI Phase 1-level funding to provide a foundation for this project, and with the improved visibility that funding and a fully functional site will provide, we should be able to leverage significant numbers of undergraduate and graduate projects.

10 Results from Prior NSF Support

NSF CCLI Phase 1 Award (DUE-0127225): *Teaching Software Testing On-line.* PI: Stephen Edwards. $74,934 for 08/2002 – 08/2006. Title: This project partially funded the initial prototype of Web-CAT, and has been instrumental in its dissemination. As a direct result of this work, Web-CAT has developed into a widely used automated grading system.

NSF CCLI Phase 1 Award (DUE-0633594): *Educational Support for Testing Graphical User Interfaces.* PI: Stephen Edwards (with one Co-PI). $149,899 for 05/2007 – 05/2009. Researches effective techniques for supporting student-written tests of GUI-based Java programs.

NSF CCLI Phase 2 Award (DUE-0618663): *Community Resources for Automated Grading.* PI: Stephen Edwards (with three Co-PIs). $433,844 for 09/2006 – 08/2009. This project focuses on the technology needed to support broad-based dissemination and use of Web-CAT, together with multi-institution evaluation of its effect on student learning.

Based on the work of these projects, Dr. Edwards has produced three journal articles, six conference papers, two workshop papers, two conference posters, a CCSC-East conference tutorial, an OOPSLA tutorial, and three SIGCSE workshops. These three projects have sponsored five minority summer research interns. As a result of these efforts, Web-CAT is now in use at over 30 institutions worldwide.

NSF ITR Award (CCR-0113181): *Modular Detection of Interface Violations in Formally Specified Software Components.* Co-PI: Stephen Edwards. $412,099 for August, 2001 – August, 2004. This project developed scalable approaches to systematically detecting violations of behavioral contracts. The primary approach investigated was the use of interface violation detection wrappers. The project focused on the test oracle problem and how it can be automatically addressed.

References

[1] Animation repository. `http://www.animal.ahrgr.de/animations.php`, 2008.

[2] Association for Computing Machinery. The ACM digital library. `http://portal.acm.org`, 2008.

[3] J. Bergin, K. Brodie, M. Patio-Martnez, M. McNally, T.L. Naps, S.H. Rodger, J. Wilson, M. Goldweber, S. Khuri, and R. Jimnez-Peris. An overview of visualization: its use and design: report of the working group in visualization. In *ITiCSE '96: Proceedings of the 1st Conference on Integrating Technology into Computer Science Education*, pages 192–200, 1996.

[4] M.H. Brown and R. Sedgewick. A system for algorithm animation. In *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 177–186, 1984.

[5] M.D. Byrne, R.Catrambone, and J.T. Stasko. Do algorithm animations aid learning? Technical Report GIT-GVU-96-18, Georgia Institute of Technology, 1996.

[6] CITIDEL: Computing and information technology interactive digital educational library. `http://www.citdel.org`, 2008.

[7] Connexions scholarly content repository. `http://cnx.org`, 2008.

[8] M.L. Cooper. Algorithm visualization: The state of the field. Master's thesis, Virginia Tech, April 2007.

[9] S. Grissom, M.F. McNally, and T.L. Naps. Algorithm visualization in CS education: comparing levels of student engagement. In *SoftVis '03: Proceedings of the 2003 ACM Symposium on Software Visualization*, pages 87–94, 2003.

[10] J.S. Gurka and W. Citrin. Testing effectiveness of algorithm animation. In *Proceedings of the 1996 IEEE Symposium on Visual Languages VL '96*, pages 182–189, September 1996.

[11] S.R. Hansen, N.H. Narayanan, and D. Schrimpsher. Helping learners visualize and comprehend algorithms. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 13(3):291–317, 2000.

[12] Hope College. Complete collection of algorithm visualizations. `http://www.cs.hope.edu/~dershem/ccaa/ccaa`, 2006.

[13] C. Hundhausen and S.A. Douglas. Using visualizations to learn algorithms: should students construct their own, or view an expert's? In *Proceedings of the 2000 IEEE International Symposium on Visual Languages VL '00*, pages 21–28, September 2000.

[14] C. Hundhausen, S.A. Douglas, and J.T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13(3):259–290, 2002.

[15] D.J. Jarc, M.B. Feldman, and R.S. Heller. Assessing the benefits of interactive prediction using web-based algorithm animation courseware. In *SIGCSE '00: Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*, pages 377–381, March 2000.

[16] JERIC: Journal on Educational Resources in Computing. http://www.acm.org/pubs/jeric, 2008.

[17] A.W. Lawrence, J. Stasko, and A. Badre. Empirically evaluating the use of animations to teach algorithms. In *Proceedings of the 1994 IEEE International Symposium on Visual Languages VL '94*, pages 48–54, October 1994.

[18] T.L. Naps, S. Cooper, B. Koldehofe, C. Leska, G. Rößling, W. Dann, A. Korhonen, L. Malmi, J. Rantakokko, R.J. Ross, J. Anderson, R. Fleischer, M. Kuittinen, and M. McNally. Evaluating the educational impact of visualization. *SIGCSE Bulletin*, 35(4):124–136, 2003.

[19] T.L. Naps, J.R. Eagan, and L.L. Norton. Jhavé: An environment to actively engage students in web-based algorithm visualizations. In *SIGCSE '00: Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*, pages 109–113, 2000.

[20] T.L. Naps, G. Rössling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S.H. Rodger, and J. Ángel Velázquez-Iturbide. Exploring the role of visualization and engagement in computer science education. In *ITiCSE-WGR '02: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 131–152, 2002.

[21] T.L. Naps and B. Swander. An object-oriented approach to algorithm visualization: easy, extensible, and dynamic. In *SIGCSE '94: Proceedings of the twenty-fifth SIGCSE symposium on Computer science education*, pages 46–50, 1994.

[22] NSDL: National Science Digital Library. http://nsdl.org, 2008.

[23] Open Archives Insitute Protocol for Metadata Harvesting. http://www.openarchives.org/OAI/openarchivesprotocol.html, 2008.

[24] Keith O'Hara and Jennifer Kay. Open source software and computer science education. *J. Comput. Small Coll.*, 18(3):1–7, 2003.

[25] Open Source Initiative: The Approved Licenses. http://www.opensource.org/licenses/, 2008.

[26] S. Ponce. Algoviz graph creator. http://research.cs.vt.edu/algoviz/gds_algoviz/AlgovizGraphCreator.html, 2008.

[27] G. Rößling, T.L. Naps, M.S. Hall, V. Karavirta, A. Kerren, C. Leska, A. Moreno, R. Oechsle, S.H. Rodger, J. Urquiza-Fuentes, and J. Ángel Velázquez-Iturbide. Merging interactive visualizations with hypertextbooks and course management. *SIGCSE Bulletin*, 38(4):166–181, 2006.

[28] P. Saraiya, C.A. Shaffer, D.S. McCrickard, and C. North. Effective features of algorithm visualizations. In *SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, pages 382–386, March 2004.

[29] GNU Savannah. http://savannah.gnu.org/, 2008.

[30] C.A. Shaffer, M.L. Cooper, and S.H. Edwards. Algorithm visualization: a report on the state of the field. In *SIGCSE '07: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, pages 150–154, March 2007.

[31] SIGCSE Educational Links. `http://sigcse.org/topics`, 2008.

[32] SMETE Digital Library. `http://www.smete.org`, 2008.

[33] Sourceforge.net. `http://sourceforge.net`, 2008.

[34] Virginia Tech Data Structures and Algorithm Visualization Research Group. Algoviz wiki annotated bibliography of the AV research literature. `http://algoviz.cs.vt.edu/AlgovizWiki/AnnotatedBibliography`, 2008.

[35] Virginia Tech Data Structures and Algorithm Visualization Research Group. Data Structures and Algorithm Visualization Wiki. `http://algoviz.cs.vt.edu`, 2008.

[36] T. Waldmann and J. Hermann. MoinMoin Wiki Engine. `http://moinmo.in`, 2008.