First Steps Towards a Visualization-Based Computer Science Hypertextbook as a Moodle Module

Guido Rößling, Teena Vellaramkalayil
CS Department, TU Darmstadt
Hochschulstr. 10
64289 Darmstadt, Germany

roessling@acm.org

Abstract

Hypertextbooks for Computer Science contents present an interesting approach to better support learners and integrate algorithm animations into the learning materials. We have developed a prototype for integrating a selection of the functionality of such a hypertextbook into the established Moodle LMS. This paper describes the goals and realization of this module together with an example.

1 Introduction

Algorithm visualization (AV) has a long tradition in visually presenting dynamic contents - typically, algorithms and data structures. The discipline and its associated tools promise easier learning and better motivation for learners. However, while surveys usually show interest in AV use, the adoption of AV by educators is lower than the proponents and developers of such systems would hope and expect. In the following, we will use the term AV whenever we refer to algorithm or program visualization or animation.

In a survey performed by the ITiCSE 2002 Working Group on 'Improving the Educational Impact of Algorithm Visualization", the main reasons why educators do not use AV materials in their lectures can be reduced to two aspects: the *time* required to do so and the *lack of integration* with existing teaching materials (Naps et al., 2003).

Since that report, several approaches have addressed the *time* aspect, for example by providing tools or generators for quickly producing content that fits the educator's or learner's expectations, and allow the user to specify the input values (Rößling and Ackermann, 2006; Naps, 2005). However, the integration of AV into the learning materials still needs to be addressed.

A 2006 ITiCSE Working Group therefore proposed a combination of hypertext-based textual materials with image, video, and AV content, as well as aspects from a learning management system (Rößling et al., 2006). This combination was called a *Visualization-based Computer Science Hypertextbook (VizCoSH)* to illustrate the main aspect form the Working Group's point of view: the seamless integration of AV materials into the learning materials used for a course.

In this paper, we present our first approach of implementing a VizCoSH. As stated by previous authors of related hypertextbooks, the effort required to create a full-fledged hypertextbook is intense. The well-known theory hypertextbook *Snapshots of the Theory of Computing* (Ross, 2006; Boroni et al., 2002), while far from finished, already represents the work of about twelve years. Therefore, it seems unlikely that a full-fledged VizCoSH - including the features "borrowed" from course or learning management systems described in the Working Group Report (Rößling et al., 2006) - could already exist if it were built from scratch since 2006.

Our first prototype for a VizCoSH is based on the popular and established Moodle learning content management system (Cole and Foster, 2007). Section 2 presents the goals for the development of the modue. Section 3 describes the approach taken for meeting these goals, followed by a short demo of the resulting content pages in Moodle in Section 4. Section 5 presents a brief evaluation of the module and concludes the paper.

2 Goals for implementing a VizCoSH

The report that presented the concept of a VizCoSH set many ambitious goals for a full-fledged VizCoSH. For example, these concerned navigation, adaptation of the contents and learning paths to the user, and the integration of tracking and testing facilities to better determine the user's understanding. While most of the goals are already implemented in "some" systems, their combination - especially with the seamless integration of animations envisioned for a VizCoSH - has not been managed so far.

For the purpose of this research, we had to scale down the expectation towards a full VizCoSH to a manageable amount. Essentially, we expected that our VizCoSH prototype should offer the following features:

- Adaptation of layout (such as fonts and color settings) and language to the user's needs,
- Addition of arbitrary elements, such as text blocks, images, or hyperlinks at any position in the contents,
- Support for asking multiple-choice quizzes and performing knowledge tests,
- Support for different user roles, at least distinguishing between *teacher* and *student*,
- Logging the user's activities, in order to be able to track individual progress,
- Basic communication features, such as chats, forums and votings,
- Structured textual elements organized similarly to a text book (otherwise, the resource could not be called a hypertextbook),
- Enabling the printing of the learning materials with about the same comfort as for a "regular" text book,
- Seamless integration of AV content at (almost) any position in the contents,
- Support for fixed AV content as well as for "random" or user-generated AV content.

Many of these goals are already addressed by a variety of software. For example, AV systems such as ANIMAL already provide the last two items in the list (Rößling and Ackermann, 2006). For most of the communication- and layout-based goals, there is a whole set of software that is geared to provide these aspects: learning content management systems including the popular *Moodle* system (Cole and Foster, 2007). We therefore decided to base our implementation on Moodle, which already offers the first six of the 10 required features.

3 Realizing a VizCoSH prototype as a Moodle module

Moodle is a highly extensible system, making it (comparatively) easy to provide additional features. The large international developer community can help in locating bugs and fixing them. However, the popularity of Moodle and the large number of developers also means that the number of offered modules or plugins for download is very large - currently, the web page lists more than 320 such elements.

The first six aspects - adaption of the visual layouts and language, management of arbitrary elements, quizzes and tests, user roles, logging, and communication features - are already integrated into Moodle and do not require further work. For the text structure similar to a book including useful printing facilities, we found a fitting "activity module" called *Book* (Škoda, 2007). This module provides the "significant structure" required by a VizCoSH, ensuring that the creation of meaningful text-based learning materials with AV content additions are possible.

The *Book* module allows printing the current "chapter" or the full book. Some limitations exist; for example, the author of the module has decided not to support sub-chapters or deeper levels of structure. Additionally, the module is not interactive, so that forums, chats etc. cannot be integrated into the content, but can be linked from anywhere in the page.

We have extended the Book module to include support for AV content and renamed it to *vizcosh*. Teachers can maintain a list of supported AV content files inside the module. New AV content can be added by providing the following information about the content: title, description, author (by default, the user currently logged in), and topic(s) covered. Additionally, the animation file has to be uploaded, optionally together with an image to be used as a thumbnail. Finally, the user has to select the animation format.

The *vizcosh* module currently supports the following formats:

- JAWAA (Akingbade et al., 2003),
- GAIGS (Naps and Rößling, 2006),
- JHAVÉ with a local file as a parameter (Naps, 2005),
- JHAVÉ with a specific input generator (Naps, 2005),
- the generators offered by Animal, where the content author can either select the generator front-end, preselect an algorithm category, or specify a specific generator (Rößling and Ackermann, 2006),
- and the internal and AnimalScript-based formats supported by Animal (Rößling and Freisleben, 2002).

Each animation format description also contains a template for starting the content with an appropriate JNLP file. The JNLP file uses a set of placeholders to substitute the actual file name etc. when it is started inside Moodle, as shown in Table 1. Note that depending on the underlying system, not all of these placeholders may be used.

Variable	**Use**
`JNLP-PATH`	Describes the base path for all relative paths used in the JNLP file, as specified by the *codebase* attribute of the JNLP specification.
`JNLP-FILENAME`	Defines the name (relative to `JNLP-PATH`) for the JNLP file, used for the *href* attribute of the JNLP root element.
`JAR-PATH`	Defines the location of the JAR file(s) for the *jar* element of the *resources* JNLP element; describes where the JAR file(s) for the AV system are to be found.
`DATA-TYPE`	Describes the type of the file, needed if the chosen system can handle more than one type of file.
`DATA-PATHFILENAME`	Specifies the path and name for a file attribute to be passed in to the chosen AV system.

Table 1: Variables used for JNLP templates

An example JNLP template for the Animal AV system is shown in Listing 1.

Listing 1: Example JNLP specification for Animal

```
<?xml version="1.0" encoding="utf-8"?>
<!-- JNLP Specification for Animal 2.3.14 distribution -->
<jnlp spec="1.0+"
    codebase="<JNLP-PATH>"
```

```
5       href="<JNLP-FILENAME>">
6     <information>
7       <title>Animal Algorithm Animation, version 2.3.14</title>
8       <vendor>Animal Developer Team / Dr. Guido Roessling</vendor>
9       <homepage href="http://www.animal.ahrgr.de" />
10      <description>Animal Algorithm Animation, v. 2.3.14</description>
11      <description kind="short">An extensible algorithm animation tool
12        used for Computer Science Education purposes</description>
13      <icon href="Animal.gif"/>
14      <icon kind="splash" href="Animal.gif"/>
15      <offline-allowed/>
16    </information>
17    <security>
18      <all-permissions/>
19    </security>
20    <resources>
21      <j2se href="http://java.sun.com/products/autodl/j2se" version="1.5+"/>
22      <jar href="<JAR-PATH>/Animal-2.3.14.jar"/>
23    </resources>
24    <application-desc>
25      <argument><DATA-TYPE></argument>
26      <argument><DATA-PATHFILENAME></argument>
27    </application-desc>
28  </jnlp>
```

Lines 4 and 5 show that the *JNLP-PATH* and *JNLP-FILENAME* describe the location of the JNLP specification file. Lines 6 to 16 provide metadata about the AV system (here, the Animal AV system), such as the title, vendor, homepage, description and icon. In line 18, all permissions are requested to allow users to save animation files to their local disk.

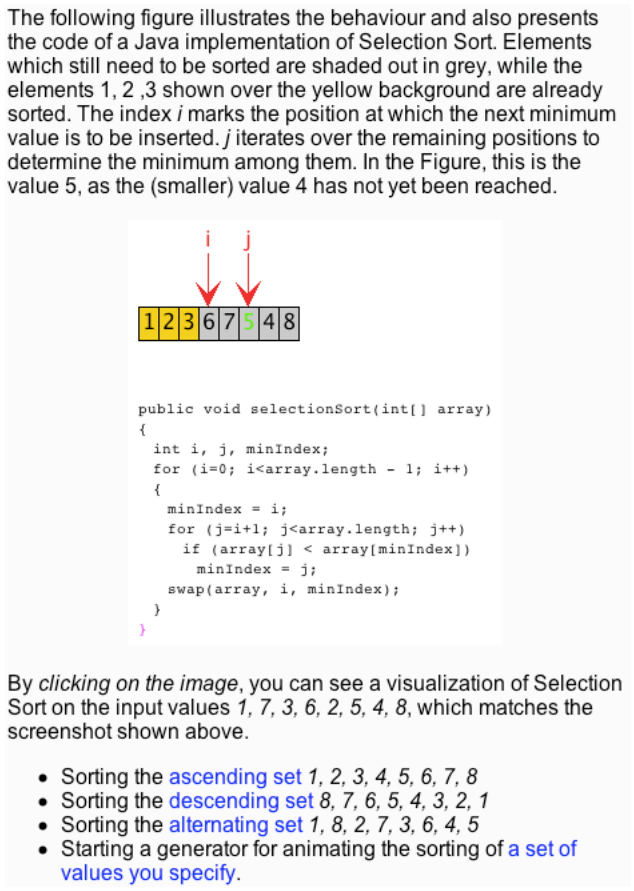
The JAR file for running the AV content is defined in line 22. Finally, lines 25 and 26 describe the run-time parameters for an animation, here the format and name of the animation file to be loaded.

Using the "Add Algorithm Visualization" button next to the standard editor, the user can easily add AV content at the end of the text. To do so, he or she simply chooses one of the existing elements from the AV content list, or creates a new entry. If the link appears in the wrong place, it can easily be cut and pasted to the right target position.

4 Example Output

Figure 1 shows an excerpt of a VizCoSH page created using the Moodle module. On this page, the image in the center is a link to the concrete visualization of the underlying sorting algorithm (here, Selection Sort). When the user clicks on this image, the AV system is started and shows the content indicated by the thumbnail. Additionally, a set of links to different alternative AV content are placed in the item list below the image, including the possibility for the user to adapt the content to his own preferences.

To keep the Figure readable, we present only a segment without the navigation elements placed above, below, and to the left of the page contents. The page also contains a paragraph describing the algorithm (Selection Sort) above Figure 1, as well as a paragraph about the complexity of the algorithm. Additionally, a number of exercises are also put on the page. In a future version of the VizCoSH, the user shall also be able to submit a solution to these exercise tasks for (semi-)automatic grading. However, this part of the module does not exist yet. We mention it to illustrate why a VizCoSH can be so much more than a "simple" text book. A "complete" VizCoSH can incorporate automatically evaluated tests that may also have a effect on how further content is presented to the user - or even which content will be visible.



The following figure illustrates the behaviour and also presents the code of a Java implementation of Selection Sort. Elements which still need to be sorted are shaded out in grey, while the elements 1, 2 ,3 shown over the yellow background are already sorted. The index *i* marks the position at which the next minimum value is to be inserted. *j* iterates over the remaining positions to determine the minimum among them. In the Figure, this is the value 5, as the (smaller) value 4 has not yet been reached.

```
public void selectionSort(int[] array)
{
  int i, j, minIndex;
  for (i=0; i<array.length - 1; i++)
  {
    minIndex = i;
    for (j=i+1; j<array.length; j++)
      if (array[j] < array[minIndex])
        minIndex = j;
    swap(array, i, minIndex);
  }
}
```

By *clicking on the image*, you can see a visualization of Selection Sort on the input values *1, 7, 3, 6, 2, 5, 4, 8*, which matches the screenshot shown above.

- Sorting the ascending set *1, 2, 3, 4, 5, 6, 7, 8*
- Sorting the descending set *8, 7, 6, 5, 4, 3, 2, 1*
- Sorting the alternating set *1, 8, 2, 7, 3, 6, 4, 5*
- Starting a generator for animating the sorting of a set of values you specify.

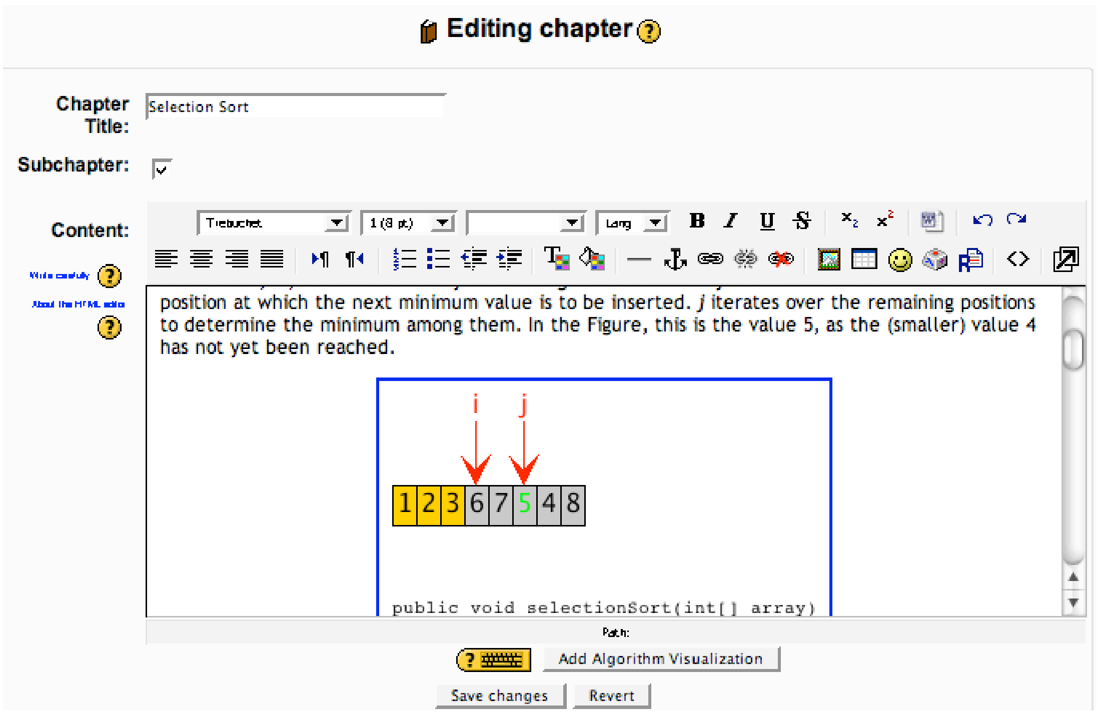
Figure 1: VizCoSH example from the *vizcosh* Moodle module

Figure 2 shows an example of the (modified) content editor provided by the *vizcosh* module. The WYSIWYG editor is already provided by Moodle. Our addition to this editor is the button *Add Algorithm Visualization* shown next to the keyboard icon. When the user clicks on this button, he is led to the collection of all registered animations. By selecting one of these entries, the associated thumbnail or text is inserted into the text editor - the AV content is now ready to be run as soon as the changes have been saved.

If the user wants to add an algorithm visualization to the VizCoSH, he clicks on a "+" icon above the list of known AV content. He is then led to the page shown in Figure 3. Here, the user can enter the title and description of the AV content. The "Author" field is automatically set to the name of the user currently logged in to Moodle. The "Format" list displays a set of predefined formats. Each format has a proper JNLP file that is automatically adapted to run the new animation, based on the animation file uploaded by the user and the JNLP features. Finally, the user can decide to use the default thumbnail , download a new one, or instead display only a text for the hyperlink. By pressing the "Create" button, the file is uploaded and placed into the proper directory, and a new JNLP file will be created that fits the AV content.

5 Evaluation and Future Work

Using the *vizcosh* Moodle module presented in this paper, it is easy to incorporate animations of any of the supported types listed in Section 3. The average time effort for adding a new



Editing chapter

Chapter Title: Selection Sort

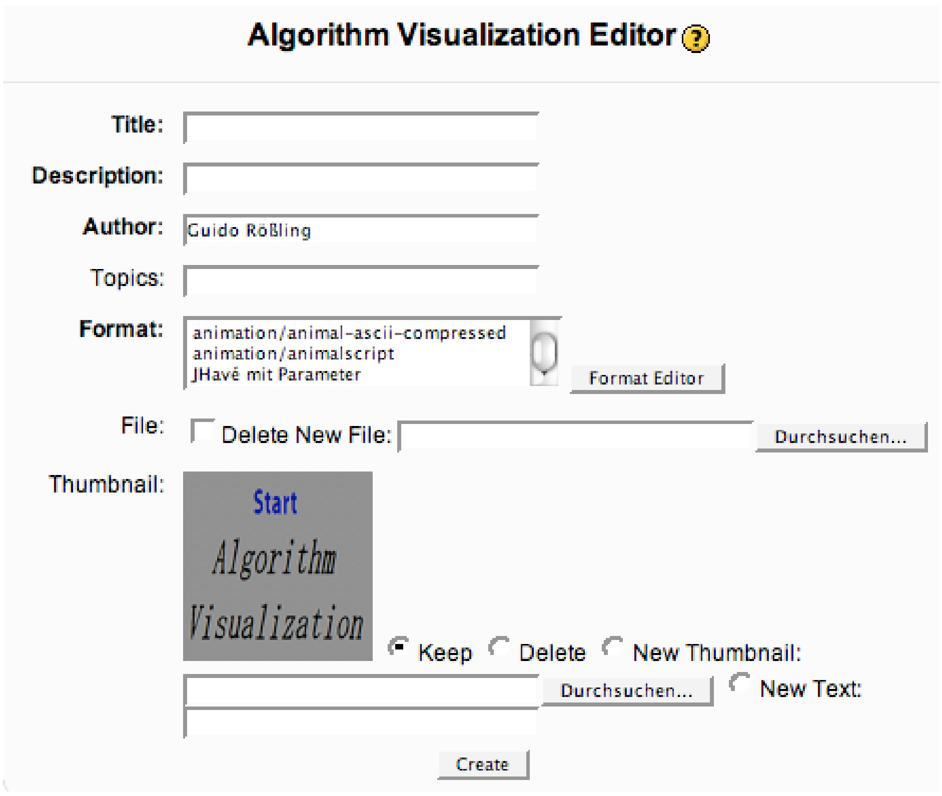
Subchapter: ✓

Content:

position at which the next minimum value is to be inserted. *j* iterates over the remaining positions to determine the minimum among them. In the Figure, this is the value 5, as the (smaller) value 4 has not yet been reached.

```
public void selectionSort(int[] array)
```

Figure 2: VizCoSH example from the *vizcosh* Moodle module



Algorithm Visualization Editor

Title:	
Description:	
Author:	Guido Rößling
Topics:	
Format:	animation/animal-ascii-compressed animation/animalscript JHavé mit Parameter
File:	Delete New File:
Thumbnail:	[✓ Keep] Delete New Thumbnail: New Text:

Figure 3: VizCoSH example from the *vizcosh* Moodle module

visualization to a given module page is less than two minutes, as it only requires creating a new animation entry and selecting the proper animation format.

At the moment, our module only supports contents that are based on a scripting notation: JAWAA, GAIGS, JHAVÉ and Animal. This does not mean that the approach is in any way restricted to scripting input - it would be just as easy to support animation systems that use a stored file of some type. However, as we are most familiar with the listed systems, there is currently no example for the support of other systems. There are real "technical" reasons for this lack of support. The main task to be done is to provide a working JNLP template for a new system, similar to the one shown in Listing 1. The author can create a new format template with a few mouse clicks by opening the "Format Editor" using the button next to the Format list shown in Figure 3. However, the basic JNLP file has to be edited manually to adapt it to the target format. This especially concerns the *application-desc* element of the JNLP specification, which states the main class of the JAR file(s) and the invocation arguments. The module itself does not have to be changed if a new format is introduced.

The ability to print the current chapter including the thumbnails is also helpful. Here, the underlying *Book* module simply renders the page content(s) without the navigation elements and redisplays them as a Web page in a new browser window. Of course, the dynamic visualization elements are reduced to static images in this approach.

Our prototype can only represent the first step towards implementing a VizCoSH. Still, we expect that it will be easy to use for others once we publish the module, and may make adoption of AV easier. Future work for the module includes a seamless incorporation of interactive features, such as Moodle's forums and chat abilities with the page content and the visualizations. For example, users should be able to link to a given visualization easily in all other areas of Moodle. User tracking for performance in built-in knowledge tests would be another important addition.

We are interested in cooperating with researchers and teachers who want to use the module. This especially concerns the support for other algorithm animation or visualization systems that can be run using Java Webstart. Several persons may not use Moodle, but some other platform, for example due to a university-enforced policy. Most of the module is not specific to Moodle and should be easy to carry over to other learning content management systems or content management systems, such as *Drupal, Plone* or *Typo3*. The main Moodle-specific aspect is the connection to the user management and database, which has to rewritten for each target platform.

We are also looking for partners who could contribute content, and allow us to take one step further towards the vision of a "community-shared" VizCoSH, as anticipated by the ITiCSE 2006 Working Group (Rößling et al., 2006).

References

Ayonike Akingbade, Thomas Finley, Diana Jackson, Pretesh Patel, and Susan H. Rodger. JAWAA: Easy Web-Based Animation from CS 0 to Advanced CS Courses. In *Proceedings of the 34th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada*, pages 162–166. ACM Press, New York, 2003.

Christopher Boroni, Frances Goosey, Michael Grinder, and Rockford Ross. Active Learning Hypertextbooks for the Web. *Journal of Visual Languages and Computing*, 13(2):341–354, 2002.

Jason Cole and Helen Foster. *Using Moodle: Teaching with the Popular Open Source Course Management System*. O'Reilly, 2007. ISBN 978-0596529185.

Thomas Naps. JHAVÉ – Addressing the Need to Support Algorithm Visualization with Tools for Active Engagement. *IEEE Computer Graphics and Applications*, 25(6):49–55, December 2005.

Thomas L. Naps and Guido Rößling. JHAVÉ - more Visualizers (and Visualizations) Needed. In Guido Rößling, editor, *Proceedings of the Fourth Program Visualization Workshop, Florence, Italy*, pages 112–117, June 2006.

Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and J. Ángel Velázquez-Iturbide. Exploring the Role of Visualization and Engagement in Computer Science Education. *ACM SIGCSE Bulletin*, 35(2):131–152, June 2003.

Rockford J. Ross. Snapshots of the theory of computing. Available online at `http://www.cs.montana.edu/webworks/projects/snapshots/`, 2006.

Guido Rößling and Tobias Ackermann. A Framework for Generating AV Content on-the-fly. In Guido Rößling, editor, *Proceedings of the Fourth Program Visualization Workshop, Florence, Italy*, pages 106–111, June 2006.

Guido Rößling and Bernd Freisleben. Animal: A System for Supporting Multiple Roles in Algorithm Animation. *Journal of Visual Languages and Computing*, 13(2):341–354, 2002.

Guido Rößling, Thomas Naps, Mark S. Hall, Ville Karavirta, Andreas Kerren, Charles Leska, Andrés Moreno, Rainer Oechsle, Susan H. Rodger, Jaime Urquiza-Fuentes, and J. Ángel Velázquez-Iturbide. Merging Interactive Visualizations with Hypertextbooks and Course Management. *SIGCSE Bulletin inroads*, 38(4):166–181, December 2006.

Petr Škoda. *book* module for Moodle. `http://docs.moodle.org/en/Book_module`, 2007.