Integration von Algorithmenvisualisierungen in die Lehre

Guido Rößling, Teena Vellaramkalayil
Rechnerbetriebsgruppe, Fachbereich Informatik
Technische Universität Darmstadt
Hochschulstr. 10
64289 Darmstadt

guido@rbg.informatik.tu-darmstadt.de
tvellaramkalayil@gmx.net

Abstract: Algorithmen und Datenstrukturen sind ein zentraler Bestandteil der Informatik. Das Verständnis wird aber durch die dynamische Natur der Verfahren erschwert. In diesem Beitrag soll gezeigt werden, wie die Vermittlung solcher dynamischer Inhalte für Dozenten und Studierende erleichtert und in die restlichen Lehrmaterialien eingebettet werden kann.

1 Einleitung

Ein wesentlicher Teil des Studiums der Informatik befasst sich mit Algorithmen und Datenstrukturen in ihren verschiedenen Ausprägungen. Dies betrifft zunächst die reine Vermittlung der grundlegenden Algorithmen und Datenstrukturen, wie sie im Rahmen des Grundstudiums Informatik erfolgt. Zahlreiche weitere Veranstaltungen setzen dieses Wissen später voraus und erweitern es um neue Verfahren und Strukturen, etwa im Bereich der verteilten Algorithmen. Gleichzeitig müssen die Studierenden hinreichend mit den Strukturen vertraut sein, um diese in der alltäglichen Programmierpraxis korrekt und effizient einsetzen zu können.

Vielen Studierenden fällt das Verstehen der Inhalte schwer. Neben dem dafür erforderlichen Abstraktionsgrad ist ein Grund, dass es sich um hochgradig dynamische Inhalte handelt. Ein Verfahren wie Quicksort oder eine Struktur wie einen AVL-Baum wird man anhand der formalen Definitionen nicht hinreichend erfassen können, wenn man nicht versteht, wie sich diese in einem konkreten Fall verhalten. Dieser Dynamik der Inhalte steht die klassischerweise eher statische Präsentation der Inhalte gegenüber, etwa auf Basis von Momentaufnahmen als „vorher-nachher“ Paar. Diese statische Vermittlung kann bereits prinzipbedingt die dynamischen Inhalte nur begrenzt darstellen. Es drängt sich also eine ebenfalls dynamische Präsentation der Inhalte auf.

Seit etwa 25 Jahren wird im Gebiet „Algorithmenanimation“ als Unterbereich der Softwarevisualisierung an geeigneten Verfahren zur Vermittlung der dynamischen Inhalte geforscht. In diesem Beitrag soll aufgezeigt werden, wie solche dynamischen Visualisierungen in der Lehre eingesetzt und direkt in die Lehrmaterialien eingebettet werden können.

Im nächsten Abschnitt werden dazu die Grundkonzepte der Algorithmenanimation dargestellt. Anschließend werden Szenarien für den Einsatz von Algorithmenanimation in der Lehre aufgezeigt.

2 Grundkonzepte der Algorithmenvisualisierung und -animation

Seit der grundlegenden Dissertation von Marc H. Brown [Bro98], ausgezeichnet als *ACM Distinguished Dissertation*, wurde an vielen Ansätzen zur Visualisierung oder Animation von Algorithmen, Datenstrukturen und Programmen geforscht.

Prinzipiell wird in dem Forschungsgebiet zwischen *Visualisierung* und *Animation* einerseits sowie zwischen *Algorithmen* und *Programmen* als Inhalten andererseits unterschieden. Insgesamt entstehen damit vier verschiedene Bereiche, die sich mit der Visualisierung bzw. Animation von Algorithmen bzw. Programmen befassen. Zur besseren Abgrenzung werden die Begriffe im Folgenden kurz definiert.

Visualisierungen beschäftigen sich vorwiegend mit einer Abbildung des Systemzustands als Abfolge statischer Bilder. Im Gegensatz dazu setzen *Animationen* auf dynamische Übergänge, bei denen Elemente nicht einfach „erscheinen“, sondern bewegt angezeigt werden. So ändern etwa bei einer Vertauschung in einem Feld die ausgewählten Elemente über eine Abfolge von Zwischenpositionen die Position, „fliegen“ also sozusagen von der alten zur neuen Position.

Einerseits könnte man an der Animation kritisieren, dass sie nicht der Realität von Zuweisungen entspricht; andererseits sorgt die zeitbehaftete Bewegung dafür, dass das auf Bewegungswahrnehmung konditionierte Auge die Änderung deutlich wahrnimmt. Damit fällt es dem Betrachter leicht zu erkennen, was sich gegenüber dem vorherigen Zustand geändert hat.

Die Unterscheidung zwischen der Darstellung von *Programmen* und *Algorithmen* trennt nicht nur prinzipielle Bereiche, sondern ganze Vorgehensweisen. Bei dem Fokus auf Programmen finden sich Systeme, die die Ausführung von Programmen repräsentieren, indem zumeist die einzelnen Befehle des Programms ausgeführt und „passend“ dargestellt werden. Im Fokus steht hier also die detailgetreue Wiedergabe des Verhaltens, wie man sie etwa von einem Debugger kennt - allerdings mit einem starken Fokus auf eine didaktisch aufbereitete Anzeige.

Bei der Darstellung von *Algorithmen* stehen hingegen konkrete Verfahren oder deren Implementierung im Vordergrund. Es muss sich dabei nicht um konkrete Algorithmen im engsten Sinne handeln; auch die Darstellung des Verhaltens von Datenstrukturen - das ja ebenfalls algorithmisch definiert ist - fällt in diesen Bereich. Im Fokus steht dabei nicht so sehr die exakte Wiedergabe der einzelnen Codezeilen, sondern vielmehr die didaktisch aufbereitete Darstellung des Ablaufs des Verfahrens.

Programm-basierte Ansätze können in der Regel nur mit einer konkreten Sprache arbeiten, die sie interpretieren. Sie besitzen meist kein Verständnis über die Intention des Codes und können diesen daher oft nicht „ideal“ visuell repräsentieren. So kann eine Struktur *X*, die

einen Wert und zwei Verweise auf Strukturen vom Typ *X* besitzt, ein Element einer doppelt verketteten Liste oder eines Binärbaums sein - oder mit beiden nichts zu tun haben. Eine Programm-basierte Darstellung muss sich in diesem Fall für eine Darstellung entscheiden; bei Algorithmus-basierten Ansätzen kann der Autor hingegen vorgeben, wie die Elemente angezeigt werden sollten.

Der Fokus dieses Beitrags liegt auf der Animation von Algorithmen. Aus klassischen und sprachlichen Gründen wird diese ebenfalls als *AV* abgekürzt, zumal die dynamische Animation eine Unterklasse der allgemeinen Visualisierung darstellt. Gegenüber der Programmvisualisierung oder -animation, wie sie in Systemen wie *Jeliot* [MMSBA04] oder *Alice* [CDP03] eingesetzt wird, kann sich der Autor bei Algorithmenanimation auch dafür entscheiden, bestimmte Aspekte bewusst abstrakter darzustellen oder zu überspringen. So wählt der Algorithmus von Dijkstra zur Suche der kürzesten Wege in einem Graph immer den *nächsten günstigsten, noch nicht besuchten Knoten*. In der Implementierung wird hier oft eine Priority Queue verwendet, die das Verfahren für Einsteiger komplexer erscheinen lässt. In einer Algorithmenanimation hat der Autor dagegen die Möglichkeit, den gewählten Knoten einfach grafisch hervorzuheben - und damit vollständig von der (komplexen) zugrunde liegenden Implementierung zu abstrahieren.

Zu den bekanntesten Vertretern der Algorithmenvisualisierung und -animation (AV) zählen *JAWAA2* [AFJ+03], *JHAVÉ* [Nap05] sowie das deutsche System ANIMAL [RF02]. *JAWAA2* unterstützt mehrere Datenstrukturen der Informatik, ist aber durch die Implementierung als Applet und die fehlende Weiterentwicklung nur noch bedingt empfehlenswert. *JHAVÉ* stellt eine Umgebung für Algorithmenanimations- und -visualisierungswerkzeuge bereit, die derzeit unter anderem von ANIMAL genutzt wird.

Im Folgenden wird nur das Algorithmenanimationssystem ANIMAL weiter betrachtet, da die Autoren damit am besten vertraut sind und das System derzeit wohl international die größte Bandbreite an unterstützten Ansätzen bietet. So ist ANIMAL das einzige aktuelle System, das die Erstellung von Inhalten sowohl in einer grafischen Benutzerschnittstelle [RF02], über manuell oder automatisch erstellte Skriptdateien [RGJW04], mittels einer Java-API [RMP07] sowie über eine große Anzahl eingebauter Animationsgeneratoren [RSK07] erlaubt. Zusätzlich kann ANIMAL per Menüauswahl jederzeit dynamisch zwischen den aktuell unterstützten Sprachen Englisch, Deutsch und Italienisch umgeschaltet werden, was sich auch auf die Elemente der grafischen Benutzerschnittstelle auswirkt.

In Abbildung 1 ist das Animationsfenster von ANIMAL zu sehen. Im oberen Bereich kann über jeweils einen Schieberegler die Ablaufgeschwindigkeit und die Vergrößerung eingestellt werden. Unten befinden sich die Kontrollelemente zum Ablaufen der Animation vorwärts oder rückwärts sowie dem „Kiosk-Modus“, bei dem alle Animationsschritte der Reihe nach wie in einem Film ablaufen. Der Schieberegler rechts unten in der Abbildung schließlich gibt den prozentualen Gesamtfortschritt der Animation wieder. Durch Ziehen des Reglers mit der Maus kann der Benutzer ebenfalls schnell durch die Animation laufen oder einzelne Punkte anspringen. Über dem Animationsfenster befindet sich ein separates Fenster, dass die einzelnen Abschnitte der Animation anzeigt und das direkte Anspringen der ausgewählten Abschnitte erlaubt. Dies setzt allerdings voraus, dass der Autor der Animation die Abschnitte entsprechend markiert und mit einem Titel versehen hat.

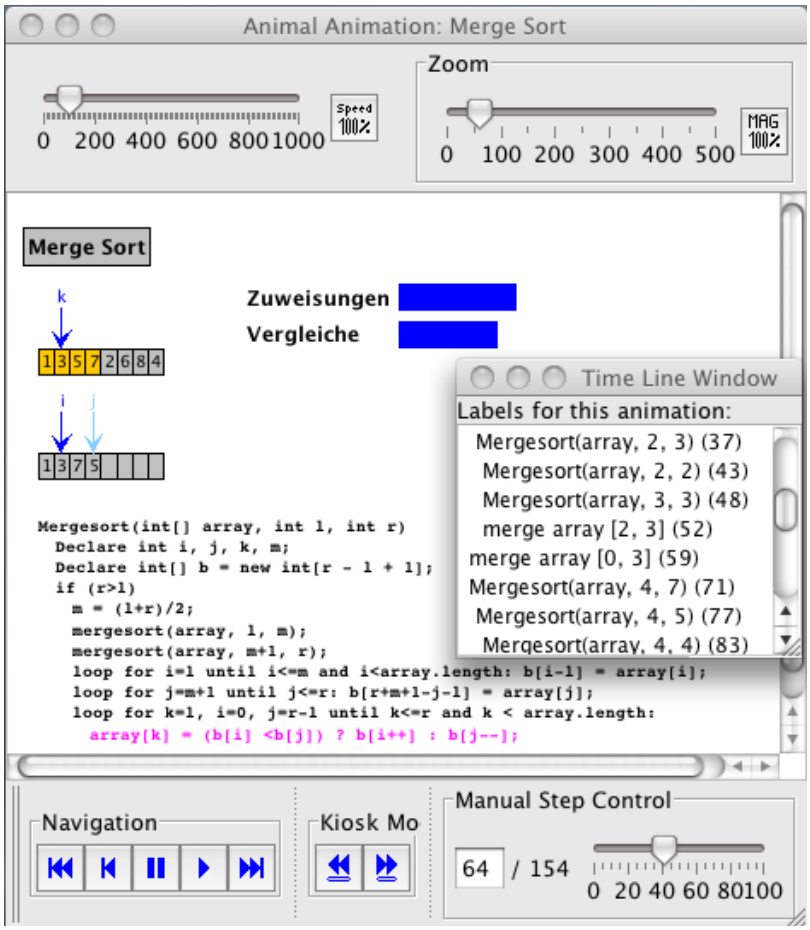


Abbildung 1: Animationsfenster von ANIMAL [RF02]

3 Integration von Algorithmenvisualisierungen in die Lehre

Angesichts einer großen Anzahl verfügbarer AV-Systeme erscheint es verwunderlich, dass diese kaum Einzug in die Lehre gefunden haben. Im Rahmen einer Untersuchung wurden Lehrkräfte zu ihrer Einstellung zum Thema befragt. Dabei stellte sich heraus, dass die Mehrzahl von den Vorteilen des Einsatzes überzeugt war, aber die zu investierende Zeit scheute, Materialien zu erstellen oder anzupassen [NRA+03]. Gleichzeitig ist die Einstellung des Lehrenden zum Einsatz der Software ähnlich entscheidend wie die Einbettung in die sonstigen Lehrmaterialien [LBA07].

Hinsichtlich der Zeit, die für die Erstellung oder Anpassung von Inhalten erforderlich ist, wurden bereits entscheidende Fortschritte erzielt. So erlauben die in ANIMAL eingebauten

Generatoren eine einfache und flexible Erstellung von Algorithmenanimation mit lediglich vier Mausklicks [RAK06]. Dabei kann der Nutzer das Aussehen sowie die zugrunde liegenden Werte der Animation des gegebenen Algorithmus direkt beeinflussen. Dazu sind lediglich in einer Eingabemaske die Farben und sonstigen Einstellungen einerseits und die konkreten Parameter - etwa das zu sortierende Feld - andererseits anzugeben.

3.1 VizCoSH: Integration von Lehrmaterialien

Die Einbettung der AV-Inhalte in die Lehre erweist sich als schwierig. Ein System, das neben den normalen Lehrmaterialien steht und mit diesen nicht passend integriert ist, wird weniger Akzeptanz und Interesse finden [LBA07]. Falls der Lehrende nicht gleichzeitig Autor des AV-Systems ist, ist eine weitgehende Anpassung der Inhalte an die Lehrmaterialien - statt anders herum - wiederum schwierig.

Vor diesem Hintergrund hat eine Forschergruppe das Konzept des *Visualization-based Computer Science Hypertextbooks* (*VizCoSH*) entwickelt [RNH+06]. Ein VizCoSH basiert auf einem klassischen Textbuch auf Hypertextbasis, stellt aber weitere Möglichkeiten bereit. So werden Aspekte von Learning Management Systemen sowie Learning Content Management Systemen wie etwa *moodle* [CF07] einbezogen, ebenso wie Aspekte adaptiver Hypermedia-Systeme. Als Ergebnis soll ein VizCoSH in der Lage sein, „klassische“ Lehrmaterialien in digitaler Form anzubieten, dabei aber auch die Inhalte an die Wünsche oder Bedürfnisse der Lerner anzupassen [BC99]. Die Einbeziehung der Funktionalitäten eines Learning Content Management Systems erleichert die Bereitstellung der Materialien sowie das Erfassen der tatsächlichen Leistungen des einzelnen Nutzers, und ermöglicht damit erst die individuelle Anpassung der Inhalte.

Von einem VizCoSH wird erwartet, dass es im Wesentlichen textbasiert ist und damit als Lehrmaterial für eine Lehrveranstaltung dienen kann. Entsprechend muss es auch eine inhaltliche Struktur aufweisen, zu der auch ein Inhaltsverzeichnis und eine „geeignete“ Gliederung in Seiten oder Kapitel zählt. Der Hauptaspekt eines solchen Hypertextbuches ist aber, dass an allen passenden Stellen auf eine in die Lehrmaterialien integrierte Algorithmenanimation oder -visualisierung verwiesen wird. Dadurch sind die visuellen Inhalte integraler Bestandteil der Lehr- und Lernmaterialien.

Zusätzlich soll auf die verschiedenen Stufen der aktiven Einbeziehung der Nutzer geachtet werden, wie diese in der “*Engagement Taxonomy*” [NRA+03] definiert wurden:

No viewing setzt keinerlei Visualisierung ein.

Viewing nutzt Visualisierungen nur passiv, indem der Nutzer lediglich den Ablauf ähnlich zu einem DVD-Wiedergabegerät steuern kann.

Responding erwartet, dass der Nutzer zu gegebenen Punkten inhaltliche Fragen zu den gezeigten oder als nächstes zu erwartenden Elementen beantworten soll. Diese Nachfragen sollen die aktive Auseinandersetzung mit den Inhalten fördern; typisch ist etwa die Frage nach dem nächsten Pivot-Element bei Quicksort.

Changing fordert den Nutzer auf, die Werte für den Algorithmus anzupassen, meist um ein konkretes Ziel zu erreichen. Dies kann etwa die Eingabe eines Arrays sei, das zum Eintreten des *worst case* bei Quicksort führt.

Constructing erlaubt es dem Nutzer, selbst aktiv eine Algorithmenanimation oder -visualisierung zu erstellen. Die intensive inhaltliche Auseinandersetzung mit einer „passenden" didaktischen Darstellung verspricht hier Lerneffekte beim Ersteller.

Presenting schließlich beschreibt den Einsatz einer Algorithmenanimation im Rahmen eines Vortrags, um Hörern einen gegebenen Sachverhalt zu vermitteln. Dabei gilt es als sekundär, ob der Vortragende die Inhalte selbst erstellt hat oder nicht.

Durch die einfache Integration von Algorithmenanimationen in das Lehrmaterial sollen die Nutzer motiviert werden, von der untersten Stufe (*No Viewing*) zumindest zur Stufe *Viewing*, also dem Einsatz von Algorithmenanimationen, zu gelangen. Von jeder höheren Stufe der aktiven Einbeziehung werden bessere Lernchancen erwartet [NRA+03].

Ein VizCoSH bietet eine direkte Integration der AV-Inhalte mit den sonstigen Lehrmaterialien. Daher ist hier zu erwarten, dass die höheren Stufen aktiver Einbeziehung leichter zu erreichen sind. So können die in der Visualisierung gestellten Fragen (Stufe *Responding*) in der dem VizCoSH zugrunde liegenden Datenbank zusammen mit den Nutzerantworten gespeichert werden. Dies erlaubt beispielsweise eine Auswertung des Lernfortschritts sowie eine optionale Anpassung der Inhalte - im einfachsten Fall in Form einer „bestanden"-Markierung.

Bei der Stufe *Changing* kann ein VizCoSH überprüfen, ob die vom Nutzer eingegebenen Werte tatsächlich die Anforderungen erfüllen, so dass der Nutzer die Aufgabe korrekt gelöst hat. Bei *Constructing* stellt das VizCoSH geeignete Werkzeuge wie das bereits erwähnte ANIMAL bereit, die die Erstellung von Visualisierungen ermöglichen. Die so erstellten Inhalte können anschließend vom Ersteller für andere Nutzer freigeschaltet werden, was einen Feedback-Prozess in Anlehnung an die Stufe *Presenting* der *Engagement Taxonomy* unterstützt.

Ein VizCoSH bietet alle Vorteile eines Lehrbuchs aus Papier und erweitert diese um einige den Lernprozess unterstützende Funktionalitäten, wie variable Navigationspfade, unterschiedliche Daten- und Inhaltsformate, elektronische Bearbeitung von Aufgaben und Tests und vor allem die Integration von Algorithmenanimationen. Diese erweiterten Funktionalitäten sollen das intensivere Auseinandersetzen des Lernenden mit dem Lernstoff durch Einbeziehung von Tests, Übungen und Algorithmenanimationen fördern. Dies sowie die Anpassung der Inhalte an den individuellen Lernstil des Lernenden (angepasste Navigationspfade, verschiedene Darstellungs- und Datenformen, ...) versprechen einen besseren Lernerfolg.

Die LMS-Elemente eines VizCoSH entlasten den Lernenden außerdem von einigen Verwaltungsaufgaben des Lernprozesses. Die zahlreichen Kommunikationswerkzeuge, die in einem VizCoSH verwendet werden können, helfen Fragen schnell zu klären und ein tieferes Verständnis zu erlangen.

Da die Verwendung eines VizCoSH den Lernerfolg eines Studenten oder Schülers steigern kann, ist es auch aus Sicht der Dozenten ein hilfreiches Mittel in der Lehre. Ergänzend

kommt hinzu, dass auch den Lehrenden zahlreiche Funktionen geboten werden, die die Verwaltung der Lehraufgaben unterstützen.

3.2 Einsatzszenarien

Der Forschungsbericht der Expertengruppe diskutiert sechs verschiedene Lernszenarien, die hier in Anlehnung an [RNH+06] wiedergegeben werden.

Die Beteiligung von weiteren Lehrkräften Die Beteiligung von Lehrkräften kann insbesondere die Erstellung oder Anpassung vorhandener Materialien betreffen. Der Bericht erwähnt auch die dynamische Zusammenstellung eines Kurses aus einem „Wissenspool“ an Seiten, so dass jede Lehrkraft den für die jeweilige Veranstaltung passenden Pfad durch die Seiten angeben könnte.

Die Unterstützung von Anfängern umfasst insbesondere die gesteigerte Interaktivität der Komponenten, insbesondere Algorithmenanimationen, und deren Integration in die Lehrmaterialien. Durch eine sukzessive Steigerung der aktiven Einbeziehung, wie in der *Engagement Taxonomy* in Abschnitt 3.1 beschrieben, wird der Übergang von Anfängern zu fortgeschrittenen Nutzern gefördert.

Die Unterstützung fortgeschrittener Nutzer erfolgt insbesondere auf den höheren Stufen der *Engagement Taxonomy*, indem die Studierenden selbst Inhalte erstellen und diese für die Diskussion und einen Peer Review anderen Nutzern zur Verfügung stellen können.

Peer Reviews können sich auf die von Studierenden oder anderen Nutzern erstellten Inhalte beziehen. Wie bei Peer Reviews üblich, kann die Diskussion unter gleichgestellten Nutzern dazu führen, dass die inhaltliche Kritik anders wahrgenommen und auch angenommen wird, als die von „Höhergestellten“[OM06]. Dabei kann es verschiedene Unterszenarien geben. So kann der Anteil der Inhalte erstellenden Studierenden ebenso variiert werden wie die Anzahl „Gutachter“ pro Animation. Der Autor einer gegebenen Animation kann den „Gutachtern“ bekannt oder unbekannt sein. Gleichzeitig kann auch die Identität der Gutachter dem Inhaltsersteller bekannt sein oder von ihm verborgen bleiben.

Programmieraufgaben zur Visualisierungserstellung Zur Vermittlung eines Verfahrens, beispielsweise *Bubble Sort*, können die Studierenden sich die Inhalte zunächst mit dem VizCoSH aneignen, indem sie auf die Lernmaterialien und den vorhandenen Pseudocode zugreifen. Durch die intensive Beschäftigung mit den mitgelieferten Animationen wird das genaue Verhalten des Algorithmus deutlich, was die spätere Implementierung ebenso erleichtert wie den Vergleich des Verhaltens der eigenen Implementierung mit der mitgelieferten Animation.

4 VizCoSH-Prototyp

Leider gibt es aktuell noch keine Umsetzung eines VizCoSH, wie sie in den vorherigen Abschnitten beschrieben wurde. Durch die reichhaltige Funktionalität wäre eine vollständige Umsetzung sehr aufwändig. Angesichts der Anzahl zu entwickelnder bzw. zu integrierender Elemente ist es nicht überraschend, dass es auch zwei Jahre nach Veröffentlichung des Konzepts noch keine Implementierung eines VizCoSH gibt. So dauerte die Entwicklung eines Hypertextbooks, das den hier aufgeführten Anforderungen lediglich teilweise entspricht, inklusive den Vorversionen bereits 1999 “mehrere Jahre” [BGG+99].

Im Rahmen einer Studienarbeit entstand ein erster Prototyp eines VizCoSH im Rahmen des LMS Moodle [CF07], der Teile der Gesamtfunktionalität abdeckt. Neben der großen Verbreitung von Moodle war für die Wahl auch ausschlaggebend, dass einige der von einem VizCoSH geforderten Eigenschaften bereits als Bestandteil oder Erweiterung von Moodle verfügbar sind, insbesondere im Bereich Tests und Kommunikation. Der Prototyp basiert auf dem *book*-Modul [Š07] für das LMS Moodle. Dieses Modul erlaubt den Aufbau einer Buch-ähnlichen Struktur in einem Moodle-Kurs und unterstützt gleichzeitig das Drucken des gesamten „Buches“ oder einzelner Abschnitte.

Im Rahmen der Studienarbeit wurde das *book*-Modul in ein *vizcosh*-Modul erweitert, das zusätzlich zu den Möglichkeiten eines Buches auch noch das Einfügen von Algorithmenanimationen unterstützt. Hierzu erstellt der Lehrende zunächst entsprechende Algorithmenanimationen oder lädt sie aus dem Internet. Anschließend werden diese in das System hochgeladen und mit dem korrekten Typ und einem Thumbnail zur schnellen Vorschau versehen. Nun können die Inhalte an einer beliebigen Stelle in den Text einer Seite eingefügt werden. An der Stelle der Algorithmenanimation sieht der Nutzer den Thumbnail. Nach Anklicken des Thumbnails wird eine maßgeschneiderte Spezifikation für Java Web-Start erstellt, mit der das für die Algorithmenanimation zuständige System gestartet wird.

Abbildung 2 zeigt den aktuellen Prototyp mit einem Auszug der englischen Seite zum Algorithmus *Sortierung durch Auswahl*. Der Thumbnail in der Mitte dient gleichzeitig zum Starten der Animation, worauf im Text auch hingewiesen wird. Gleichzeitig werden weitere Algorithmenanimationen bereitgestellt, damit die Nutzer sich einen möglichst guten Eindruck vom Verfahren machen können. Dies betrifft eine auf- sowie eine absteigende und eine alternierende Permutationen der Zahlen von 1 bis 8. Gleichzeitig können die Nutzer durch den Link zum Generator selbst die Animation des Verfahrens für von ihnen ausgewählte Daten veranlassen.

Bei den in Abbildung 2 aus Platzgründen nicht mehr sichbaren Übungsaufgaben sollen die Nutzer selbst eine zehnelementige Folge angeben, die zu dem jeweils angegebenen Verhalten führt, also etwa zu genau zwei Zuweisungen an Feldelemente. Diese Lösung kann auch für die automatische Korrektur hochgeladen werden - allerdings existiert dieses Modul noch nicht. Durch die Größenvorgaben sind einige Elemente, wie der Seitenanfang, die Navigation und die Komplexitätsbetrachtung, auf der Abbildung nicht mehr zu sehen, ebenso wie die GUI des Browsers (hier Firefox).

Wie bereits erwähnt, erlaubt das Modul auch das Ausdrucken des gesamten VizCoSH oder eines einzelnen Abschnitts. Damit können Studierende einfach eine druckbare Fassung

erstellen, die bis auf die interaktiven Elemente alle wesentlichen Inhalte umfasst.

Derzeit unterstützt das Modul die eingangs bereits erwähnten Systeme *JAWAA2* [AFJ+03], *JHAVÉ* [Nap05] sowie ANIMAL [RF02]. Während *JAWAA2* nur mit Übergabe des Namens der Animation gestartet werden kann, bieten *JHAVÉ* und ANIMAL verschiedene Startoptionen. So kann das System an sich gestartet werden, in dem der Nutzer dann autonom eine Animation auswählen kann. Zusätzlich kann der Dateiname der anzuzeigenden Animation übergeben werden. Über einen weiteren Parameter kann ebenfalls die Generation einer Animation zu einem bestimmten Algorithmus angestoßen werden. In ANIMAL kann zusätzlich auch noch die allgemeine Generatoren-Schnittstelle [RAK06] gestartet werden, so dass der Nutzer aussuchen kann, welche Art von Algorithmus er animiert haben möchte.

5 Zusammenfassung

Ein *Visualization-based Computer Science Hypertexbook* (VizCoSH) [RNH+06] ist ein Lehr- und Lernwerkzeug, das neben den reinen Lehrmaterialien zahlreiche andere Angebote zur Verbesserung der Lehre und des Lernens integriert. Hierzu zählen neben vielen Teilaspekten gängiger Learning Management Systeme oder Learning Content Management Systeme insbesondere die Algorithmenanimationen. Durch das Ansprechen unterschiedlicher Stufen aktiver Einbeziehung soll der Lernende sich aktiver mit den Inhalten auseinandersetzen und so effektiver und motivierter lernen können.

Das 2006 vorgestellte VizCoSH-Konzept ist sehr umfangreich und entsprechend schwierig vollständig umzusetzen. Der in diesem Beitrag vorgestellte erste Prototyp reicht noch bei weitem nicht an die Funktionalität eines vollwertigen Visualization-based Computer Science Hypertextbooks heran. Dennoch ist dies ein erster wichtiger Schritt in die Richtung der Entwicklung eines VizCoSH.

Durch die Abstützung auf das etablierte *Moodle*-System ist mit einer einfacheren Verbreitung des VizCoSH zu rechnen, als dies bei einer proprietären Lösung, etwa als komplette Java-Anwendung, der Fall gewesen wäre. Zudem setzen bereits zahlreiche Dozenten und Universitäten *Moodle* ein; für die Erprobung eines VizCoSH ist hier dann lediglich ein weiteres Modul zu installieren.

Derzeit befindet sich der Prototyp noch in der ersten Testphase und ist daher auch noch nicht öffentlich verfügbar. Sobald die Tests abgeschlossen sind, wollen wir auch eine Evaluation der Nutzung des VizCoSH durchführen. In diesem Rahmen sollen sowohl Nutzer zu ihren Eindrücken beim Erstellen oder Anpassen von Inhalten als auch Studierende bei der Nutzung des VizCoSH befragt werden. Von den Evaluationsergebnissen erhoffen wir uns Aufschluss über die nächsten Erweiterungen oder Anpassungen zu gewinnen.

Als erster Einsatz in der Lehre wird der Prototyp im Sommersemester 2008 in zwei Praktika eingesetzt, die sich der Lehrerausbildung beziehungsweise der Animation von Algorithmen widmen. Von diesem Einsatz erhoffen wir uns auch weitere Erkenntnisse über die Stärken und Schwächen des aktuellen Prototyps.

Literatur

[AFJ+03] Ayonike Akingbade, Thomas Finley, Diana Jackson, Pretesh Patel und Susan H. Rodger. JAWAA: Easy Web-Based Animation from CS 0 to Advanced CS Courses. In *Proceedings of the 34^{th} ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada*, Seiten 162–166. ACM Press, New York, 2003.

[BC99] Peter Brusilovsky und David W. Cooper. ADAPTS: Adaptive hypermedia for a Web-based performance support system. In *Proceedings of the Second Workshop on Adaptive Systems and User Modeling on the World Wide Web, Toronto, Canada*, Seiten 41–47, 1999.

[BGG+99] Christopher M. Boroni, Frances W. Goosey, Michael T. Grinder, Jessica L. Lambert und Rockford J. Ross. Tying it all together: creating self-contained, animated, interactive, Web-based resources for computer science education. *SIGCSE Bull.*, 31(1):7–11, 1999.

[Bro98] Marc H. Brown. A Taxonomy of Algorithm Animation Displays. In John Stasko, John Domingue, Marc H. Brown und Blaine A. Price, Hrsg., *Software Visualization*, Kapitel 3, Seiten 35–42. MIT Press, 1998.

[CDP03] Stephen Cooper, Wanda Dann und Randy Pausch. Introduction to OO: Teaching Objects-First in Introductory Computer Science. In *Proceedings of the 34^{th} ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada*, Seiten 191–195. ACM Press, New York, 2003.

[CF07] Jason Cole und Helen Foster. *Using Moodle: Teaching with the Popular Open Source Course Management System*. O'Reilly, 2007.

[LBA07] Ronit Ben-Bassat Levy und Mordechai Ben-Ari. We Work So Hard and They Don't Use It: Acceptance of Software Tools by Teachers. In *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, Seiten 246–250, New York, NY, USA, 2007. ACM.

[MMSBA04] Andrés Moreno, Niko Myller, Erkki Sutinen und Mordechai Ben-Ari. Visualizing Programs with Jeliot 3. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI 2004), Gallipoli (Lecce), Italy*, Seiten 373–380. ACM Press, New York, Mai 2004.

[Nap05] Thomas Naps. JHAVÉ – Addressing the Need to Support Algorithm Visualization with Tools for Active Engagement. *IEEE Computer Graphics and Applications*, 25(6):49–55, Dezember 2005.

[NRA+03] Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger und J. Ángel Velázquez-Iturbide. Exploring the Role of Visualization and Engagement in Computer Science Education. *ACM SIGCSE Bulletin*, 35(2):131–152, Juni 2003.

[OM06] Rainer Oechsle und Thiemo Morth. Peer Review of Animations Developed by Students. In *Proceedings of the 4^{th} Program Visualization Workshop (PVW 2006), Florence, Italy*, Seiten 39–43. University of Florence, 2006.

[RAK06] Guido Rößling, Tobias Ackermann und Simon Kulessa. Visualisierung von Algorithmen und Datenstrukturen. In Max Mühlhäuser, Guido Rößling und Ralf Steinmetz, Hrsg., *DeLFI 2006: 4. E-Learning Fachkonferenz Informatik, Darmstadt, Germany*,

number 87 in LNI Lecture Notes in Informatics, Seiten 231–242. Springer, November 2006.

[RF02] Guido Rößling und Bernd Freisleben. ANIMAL: A System for Supporting Multiple Roles in Algorithm Animation. *Journal of Visual Languages and Computing*, 13(2):341–354, 2002.

[RGJW04] Guido Rößling, Felix Gliesche, Thomas Jajeh und Thomas Widjaja. Enhanced Expressiveness in Scripting Using ANIMALSCRIPT V2. In *Proceedings of the Third Program Visualization Workshop, University of Warwick, UK*, Seiten 15–19, Juli 2004.

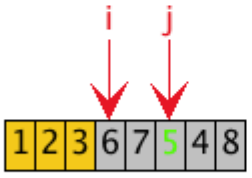
[RMP07] Guido Rößling, Stephan Mehlhase und Jens Pfau. Java-basierte Erstellung von Algorithmenanimationen. In *Proceedings der Pre-Conference Workshops der 5. e-Learning Fachtagung Informatik (DeLFI 2007)*, Seiten 77–84. Logos Verlag Berlin, 2007.

[RNH+06] Guido Rößling, Thomas Naps, Mark S. Hall, Ville Karavirta, Andreas Kerren, Charles Leska, Andrés Moreno, Rainer Oechsle, Susan H. Rodger, Jaime Urquiza-Fuentes und J. Ángel Velázquez-Iturbide. Merging Interactive Visualizations with Hypertextbooks and Course Management. *SIGCSE Bulletin inroads*, 38(4):166–181, Dezember 2006.

[RSK07] Guido Rößling, Silke Schneider und Simon Kulessa. Easy, Fast, and Flexible Algorithm Animation Generation. In *Proceedings of the* 13^{th} *ACM SIGCSE/SIGCUE International Conference on Innovation and Technology in Computer Science Education (ITiCSE 2007), Dundee, Scotland*, Seite 357. ACM Press, New York, NY, USA, 2007.

[Š07] Petr Škoda. *book* Module for Moodle. `http://docs.moodle.org/en/Book_module`, 2007.

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.

Abbildung 2: Beispielumsetzung eines VizCoSH; das Bild und die Hyperlinks führen jeweils direkt zu der entsprechenden Algorithmenanimation