Visualisierung von Algorithmen und Datenstrukturen

Guido Rößling, Tobias Ackermann, Simon Kulessa

Rechnerbetriebsgruppe, Fachbereich Informatik
TU Darmstadt
Hochschulstr. 10
64289 Darmstadt
guido@rbg.informatik.tu-darmstadt.de
TAzzu@gmx.net
skulessa@rbg.informatik.tu-darmstadt.de

Abstract: Algorithmen und Datenstrukturen zeichnen sich durch eine hohe Dynamik aus, die gerade für Studienanfänger aber oft schwer nachzuvollziehen ist. Die Animation der Inhalte soll daher das Verständnis steigern. In diesem Beitrag wird der aktuelle Stand der Forschung auf dem Gebiet präsentiert und beispielhaft illustriert, wie einfach Inhalte „passgenau“ erstellt werden können.

1 Einleitung

Ein wichtiger Teil der Informatikausbildung ist das Verstehen des Verhaltens von dynamischen Systemen, etwa Algorithmen, Datenstrukturen oder verteilten Systemen. Hierbei treten oft Probleme auf, die zum Teil direkt an den Inhalten liegen. So bereiten etwa die Doppelrotationen bei AVL-Bäumen oft Verständnisprobleme bei Studierenden im ersten oder zweiten Studienjahr. Gleichzeitig setzt die Lehre auch bei hochgradig dynamischen Systemen oft auf eher statische Medien, etwa Powerpoint-Folien mit dem Quelltext des Algorithmus sowie Diagramme oder Bilder der „Zwischenzustände“. Für die Studierenden ist es in diesen Fällen nicht leicht, einen Zugang zum tatsächlichen dynamischen Verhalten zu entwickeln.

Es bietet sich daher an, die Dynamik der Systeme ebenfalls dynamisch darzustellen. Hierzu gibt es seit etwa 20 Jahren das Forschungsgebiet „Algorithmenvisualisierung“ (AV), das zunehmend in nutzerfreundliche Systeme gemündet ist.

Durch die lange Existenz des Feldes existieren bereits mehrere konkurrierende Systeme, überwiegend auf Basis von Java. Im Folgenden werden zunächst einige dieser Systeme kurz vorgestellt, bevor auf Überlegungen zum lernfördernden Einsatz von AV sowie einfache Inhaltserstellung eingegangen wird.

2 Verwandte Arbeiten

Das Forschungsfeld Algorithmenvisualisierung (AV) arbeitet an der Erforschung, Umsetzung und Evaluation von Konzepten zur Darstellung des Verhaltens von Softwarekomponenten. Das Gesamtgebiet zerfällt in verschiedene Unterbereiche, je nachdem, ob im Vordergrund die Darstellung von Programmen – etwa auf Basis von zeilenweise interpretiertem Quelltext – steht („Programmvisualisierung“) oder die Darstellung von Algorithmen und Datenstrukturen („Algorithmenvisualisierung“). Gleichzeitig wird oft zwischen einer dynamischen Darstellung („Animation“) und einer Abfolge statischer Bilder („Visualisierung“) unterschieden. In diesem Beitrag steht die Animation von Algorithmen und Datenstrukturen im Vordergrund.

Das recht junge System *j-Algo* [Bl06] visualisiert derzeit AVL-Bäume sowie den Dijkstra-Algorithmus. Im Gegensatz zu vielen etablierten Systemen erlaubt j-Algo auch das Rückspringen in der Animation, das bei AVL-Bäumen allerdings auf den „letzten größeren Schritt“ beschränkt ist. Die Darstellung der Schritte erfolgt als Abfolge von statischen Bildern.

JAWAA2 der Duke University [Ro02] animiert Algorithmen und Datenstrukturen. Der Darstellung liegt eine Skriptsprache zu Grunde, die die Erstellung von Animationen vereinfacht. Im Gegensatz zu j-Algo erlaubt JAWAA2 dabei auch die zeitbehaftete Darstellung von Übergängen, wobei der Nutzer allerdings Dauer und Anfangszeitpunkt nicht beeinflussen kann.

Das System *Interactive Data Structure Visualization (IDSV)* [JFH00] stellt insgesamt elf verschiedene Inhalte dar, unter anderem Binärbäume, Graphen und Sortierverfahren. Neben einer kurzen Einführung in das jeweilige Thema steht eine interaktive Animation zur Verfügung. Dabei kann der Nutzer wählen, ob das System das korrekte Verhalten vorführen soll oder ob er die jeweiligen Operationen selbst versuchen will.

Das *Matrix Pro*-System [Ko03] verfolgt einen ähnlichen Ansatz, geht aber bei der Interaktion deutlich über IDSV hinaus. Das zu Grunde liegende Konzept ist eine „Übung zur Algorithmensimulation“: die Nutzer bekommen eine Aufgabe gestellt, und sollen diese möglichst identisch zu dem vorgegebenen Algorithmus bewältigen. Dabei wird der Algorithmus im Hintergrund simuliert und mit den Nutzeraktionen verglichen, so dass direkt auf Fehler hingewiesen werden kann.

Das *ANIMAL*-System [Rö04] bietet derzeit etwa 60 Animationen zu Algorithmen und Datenstrukturen. Im Gegensatz zu den anderen Systemen kann der Autor einer Animation sowohl Startzeitpunkt als auch Dauer jedes Animationseffekts definieren, was eine sehr präzise Anpassung der Animation an individuelle Wünsche erlaubt. Gleichzeitig ist es das einzige betrachtete System, das zumindest zweisprachig (Deutsch und Englisch) vorliegt, wobei weitere Sprachen durch das Übersetzen einer Sprachdatei hinzugefügt werden können. Ähnliche Konzepte zur Mehrsprachigkeit finden sich auch in *j-Algo*, sind aber noch nicht umgesetzt.

Abbildung 1 zeigt ein Beispiel einer Animation des Mergesort-Verfahrens in ANIMAL. Die obere Kontrollreihe erlaubt die Einstellung der Animationsgeschwindigkeit (0% bis 1000%) und der Vergrößerung (zwischen 0% und 500%). Für beide Einstellungen gibt es zusätzlich einen Schaltknopf zum Einstellen des Normalwertes (100%). Der aktuelle Wert wird durch den (hier nicht sichtbaren) „Tooltip“ der Schieberegler angezeigt.

Am unteren Fensterrand befinden sich die Animationskontrollen. Die Kontrollknöpfe sind symmetrisch um die Pausetaste angeordnet, da der Benutzer jede Navigation sowohl vorwärts als auch rückwärts ausführen kann. Die unterstrichenen Doppelpfeile stehen dabei für den „Kiosk-Modus“, bei dem die Animationsschritte direkt hintereinander ausgeführt werden. Die Pause zwischen zwei Schritten kann vom Nutzer definiert werden. Zusätzlich kann im Textfenster direkt eine Schrittnummer angegeben werden oder durch den Schieberegler die Animation nach vorne oder hinten „gezogen“ werden.

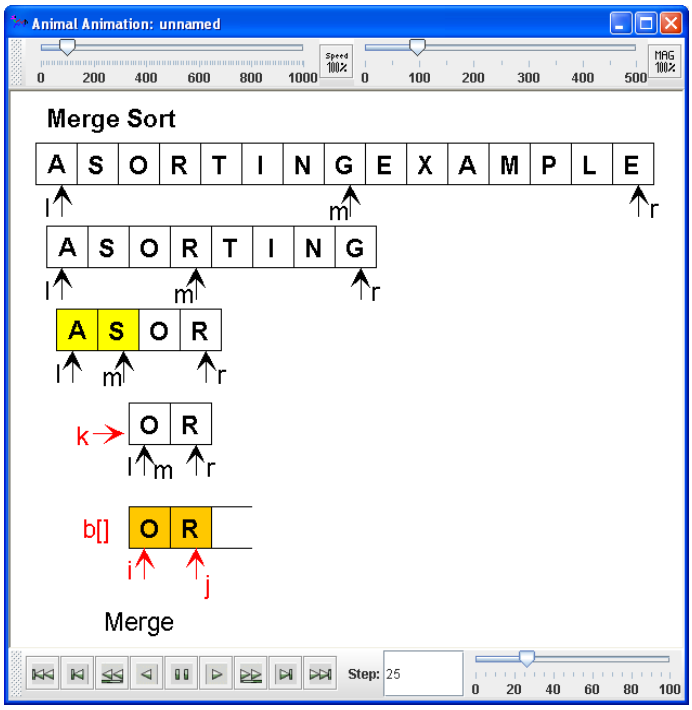

Abbildung 1: Beispiel einer Algorithmenanimation, hier Mergesort in ANIMAL

3 Problemfelder bei der Nutzung von AV-Inhalten

Eine internationale Arbeitsgruppe hat im Jahr 2002 die Haupthindernisse für eine intensivere Nutzung von AV-Technologien in der Lehre untersucht [Na02]. Die 186 Teilnehmer einer Befragung nannten als Hauptgründe dabei die Schwierigkeit, Inhalte zu finden (93%), die Nutzung des Werkzeugs zu erlernen (90%), den Zeitaufwand zur Inhaltserstellung (90%), den Mangel an geeigneten Erstellungswerkzeugen (83%) sowie den Zeitaufwand, die Inhalte an die eigenen Lehrmaterialien anzupassen (79%). Gleichzeitig ist der Beitrag der AV-Elemente zur Steigerung des Lernerfolgs aus Sicht von 59% der Befragten entweder unklar oder noch nicht hinreichend empirisch belegt.

Die Arbeitsgruppe nahm diese Problemfelder zum Anlass, allgemeine Empfehlungen für den Entwurf und die Verbreitung von AV-Inhalten auszusprechen. Für die Messung der Effektivität wurde eine „*Engagement Taxonomy*“ erstellt, die die Auseinandersetzung mit AV-Elementen in sechs Stufen einteilte:

1. keine Nutzung von AV-Elementen;
2. überwiegend passive Betrachtung von AV-Inhalten, bei der der Nutzer lediglich den Ablauf durch „Pause / weiter“ steuert;
3. Beantwortung von Fragen während der Betrachtung von AV-Inhalten;
4. Ändern der Inhalte, indem etwa für ein Problem eine „passende“ Eingabemenge bestimmt werden soll;
5. Erstellung einer neuen Animation zu einem gegebenen Inhalt;
6. Präsentation von Inhalten mittels AV vor einem Publikum.

Die Arbeitsgruppe stellte dabei zwei Hypothesen auf. Erstens wurde vermutet, dass die beiden ersten Stufen der Taxonomie keinen signifikanten Unterschied bei den Lernerfolgen bewirken: passives Betrachten führt also zu keinem besseren Lernen. Zweitens wurde vermutet, dass jede Stufe ab der dritten – bei „passender“ Umsetzung – den Lernerfolg messbar steigern kann.

Ausgehend von diesen Ergebnissen wurden einige bereits vorhandene AV-System angepasst oder erweitert, um den neuen Anforderungen besser gerecht werden zu können. So berichten etwa Rößling und Häussge [RH04] von einer systemunabhängigen Komponente zur Unterstützung von Fragen, die exemplarisch in das bereits genannte ANIMAL-System eingebettet wurde. Die bereits erwähnten Systeme *IDSV* und *Matrix Pro* stellen dem Nutzer Aufgaben während der Nutzung (äquivalent zu den Fragen von Stufe 3), erlauben aber auch das Konstruieren von Inhalten (Stufe 4). *J-Algo* unterstützt das Beeinflussen der Konstruktion durch die Angabe konkreter Werte (Stufe 4, teilweise auch Stufe 5), bietet aber keine Unterstützung für Fragen.

Neben der – noch nicht hinreichend überprüften – „Engagement Taxonomy“ liegt das Hauptproblem der Nutzung von AV-System in der meist sehr langwierigen Erstellung der Inhalte. Dabei haben sich mehrere unterschiedliche Ansätze zur Inhaltserstellung etabliert:

- Die Erstellung durch eine *grafische Schnittstelle* ist nur sehr selten anzutreffen. Sie erlaubt die größte Flexibilität bei der Inhaltserstellung und ist meist auch für Anfänger geeignet. Wie zu erwarten ist, ist die manuelle Erstellung aber auch sehr aufwändig. Von den erwähnten Systemen unterstützen nur *ANIMAL* sowie seit kurzem *JAWAA2* eine grafische Erstellung.

- Die Nutzung von *Skriptsprachen* ist recht gängig. Hierbei schreibt der Autor Textbefehle in eine Datei oder ein Eingabefenster, die die Anwendung dann parst und in entsprechende optische Effekte umsetzt. Unter den genannten Systemen findet sich dieser Ansatz wiederum nur bei *ANIMAL* und *JAWAA2*.

- Alternativ zu einer Skriptnotation kann auch eine *API* zur Erstellung von Animationen genutzt werden. Derzeit bietet von den genannten Systemen nur *ANIMAL* diese Möglichkeit. Die Nutzung von APIs beschränkt sich in der Regel auf Programmierer mit gewisser Erfahrung.

- Gelegentlich wird auch eine manuelle Erstellung durch Ausfüllen von „*Wizards*“ oder *Interaktionen mit einer GUI* angeboten. Dieser Ansatz findet sich in *IDSV*, *Matrix Pro* und *j-Algo*. Hier wird jeweils durch eine passende Interaktion des Nutzers ein neues Element hingefügt, gelöscht oder verschoben.

Aus Nutzersicht ist der letztgenannte Ansatz der schnellste und damit angenehmste. Allerdings hat er den großen Nachteil, dass nur „vorgesehene“ Inhalte erstellt werden können. Gleichzeitig ermöglichen weder *IDSV* noch *Matrix Pro* oder *j-Algo* dem Nutzer eine weitgehende Anpassung der Darstellung an seine Bedürfnisse – in der Regel ist nicht einmal die Positionierung der Elemente, etwa der Baumwurzel, möglich. Farbanpassungen oder andere Erwägungen fehlen ebenfalls. Daher ist der Ansatz insgesamt noch nicht als befriedigend anzusehen – er benötigt mehr Flexibilität und Adaptivität, um besser auf den Endnutzer eingehen zu können.

Im folgenden Abschnitt wird daher betrachtet, wie eine auf die Bedürfnisse des Nutzers anpassbare Animation erstellt werden kann, ohne gleichzeitig die dafür benötigte Zeit zu erhöhen.

3 Maßgeschneiderte schnelle grafische Erstellung von Animationen

Die Erstellung von Animationsinhalten sollte sich möglichst stark an dem orientieren, was der Nutzer erwartet oder sich wünscht. Wie oben erwähnt, empfanden 79% der Befragten es als (zu) aufwändig, die Inhalte an Ihre Vorstellungen anzupassen [Na02]. Jeweils mindestens 90% der Befragten legten großen Wert darauf, dass die Erstellung nicht zu lange dauert.

Aus diesem Grund wurde eine neue, weitgehend generische, Komponente zur Erstellung von Animationsinhalten erstellt: das *Algorithm Animation Generator Framework*. Das Framework unterstützt den Nutzer auf den (wenigen und kurzen) Schritten hin zur Erstellung von auf ihn angepassten Animationsinhalten. Im Folgenden wird dieser Ablauf näher betrachtet.

Wie in Abbildung 2 gezeigt, wählt der Nutzer zunächst die Algorithmenkategorie aus. Die gezeigte Liste umfasst die drei Kategorien *Sortieralgorithmen*, *Suchalgorithmen* und *Verschlüsselung*. Diese Liste wird beim Start des Programms dynamisch aufgebaut. Weitere Kategorien erscheinen daher automatisch, sobald es entsprechende Werkzeuge zur Inhaltserstellung gibt. Auf der rechten Seite erscheint eine kurze Beschreibung, um was es sich bei dieser Kategorie handelt.

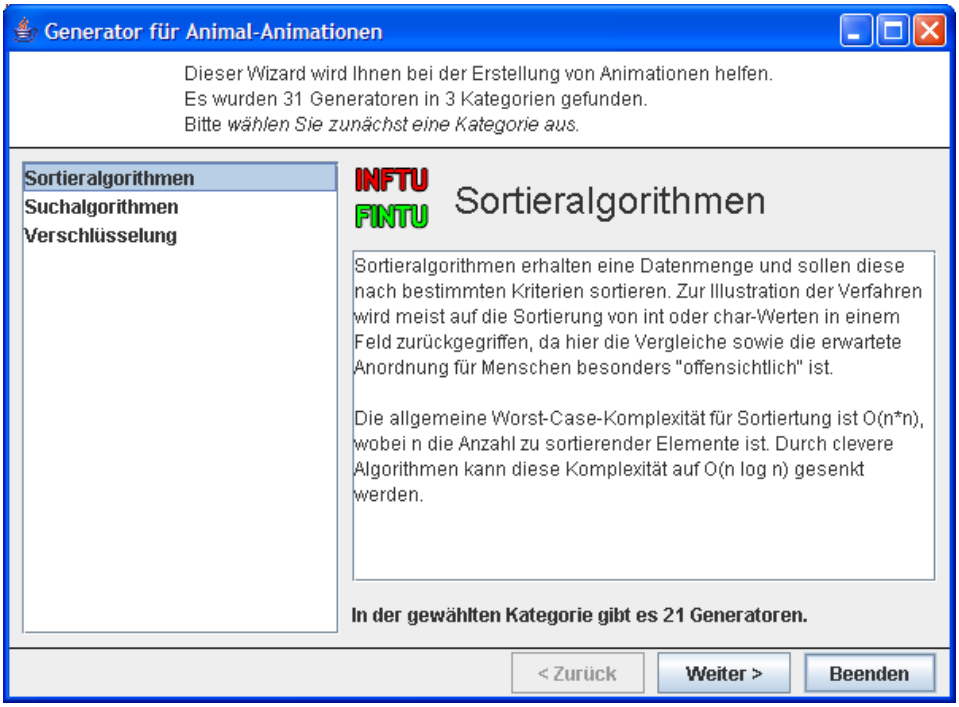

Abbildung 2: Auswahl einer Algorithmenkategorie

Anschließend wird eine Liste der verfügbaren Inhaltsgeneratoren angezeigt (siehe Abbildung 3). Der Nutzer kann wieder einen Eintrag frei auswählen. Zur besseren Einschätzung wird rechts eine Beschreibung des Generators oder des von ihm erstellten Inhalts angezeigt. Rechts unten wird der zu Grunde liegende Quelltext oder Pseudocode angezeigt. Von einigen Algorithmen gibt es eine Vielzahl an Varianten, die jeweils zu leicht unterschiedlichem Verhalten führen – und den Nutzer verwirren können, wenn sie dann nicht exakt dem erwarteten entsprechen. Das wohl bekannteste Beispiel sind hier die zahlreichen Varianten zur Wahl des Pivotelements bei Quicksort.

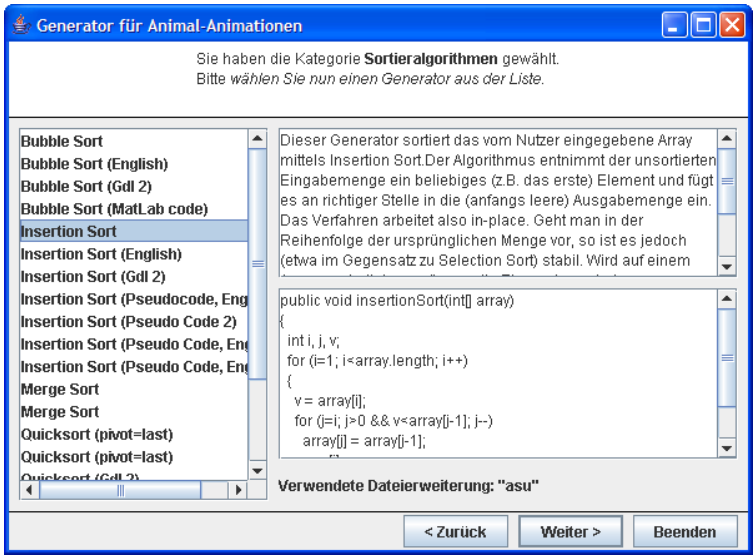

Abbildung 3: Auswahl eines Generators; rechts sind Beschreibung und Code zu sehen

Nach diesen beiden Schritten – die jeweils nur den Aufwand eines einzigen Mausklicks nach sich ziehen – kann der Nutzer nun die Eigenschaften „seiner“ Animation einstellen. Hierzu ändert sich der Inhalt des Fensters wie in Abbildung 4 gezeigt. Der Nutzer kann hier nun alle veränderlichen Werte des Generators einstellen. Im Beispiel zählen hierzu die Feldlänge, die einzelnen Feldwerte (im Beispiel wurde gerade Element 1 bearbeitet), sowie die grafische Darstellung der Elemente. Damit ist die Anpassung der Inhalte an die gewünschten Werte sowie ein „Corporate Design“ der Präsentation möglich.

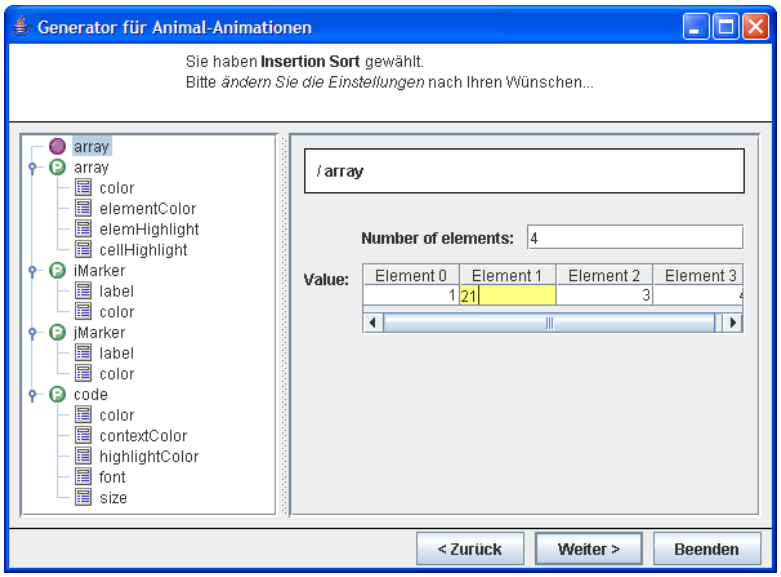

Abbildung 4: Anpassung der Werte und grafischen Darstellung an Nutzerwünsche

Zuletzt muss der Nutzer nur noch einen Dateinamen für die erstellten Inhalte angeben. Der entsprechende Generator wird nun mit den vom Nutzer angegebenen Werten gestartet und erstellt die Inhalte. Diese werden anschließend in die angegebene Datei gespeichert. Nun können die Inhalte direkt eingeladen und im entsprechenden System angezeigt werden.

Durch das Generator-Framework können Animationsinhalte mit wenigen Mausklicks erzeugt und visuell an die individuellen Wünsche des Nutzers angepasst werden. Eine weitergehende Personalisierung – etwa das Einblenden von Zusatzinformationen – erfordert eine Modifikation des Generators. Die eingangs zitierten Hauptargumente gegen den Einsatz von AV-Systemen sind damit weitgehend entkräftet: die Nutzung des Systems ist hinreichend einfach und schnell und erlaubt „maßgeschneiderte“ Ergebnisse.

Zusätzlich zu dem Generator-Framework existiert eine Komponente zur Animation von Graphen und Graphenalgorithmen. Derzeit ist diese Komponente nicht in das Generator-Framework integriert; dies ist aber für die Zukunft geplant. Abbildung 5 illustriert die Nutzung der Komponente zur Erstellung eines Graphen. Ob der Graph gerichtet und / oder gewichtet ist, wird unter dem Karteireiter „Eigenschaften“ eingestellt.

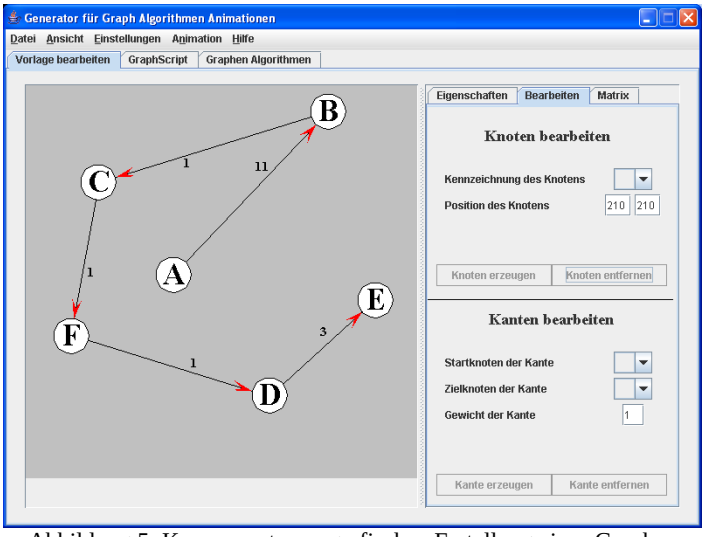

Abbildung 5: Komponente zur grafischen Erstellung eines Graphen

Die linke Seite von Abbildung 5 zeigt das Panel, auf dem die einzelnen Knoten durch Ziehen platziert werden. Rechts können die Knoten auch manuell bearbeitet werden, indem die Koordinaten direkt eingegeben werden. Darunter ist das Panel zum Erstellen von Kanten durch Auswahl des Anfangs- und Endknotens sowie des Kantengewichts zu sehen. Alternativ zur Eingabe der Kanten kann auch die Adjazenzmatrix des Graphen bearbeitet werden (Karteireiter „Matrix“ rechts oben). Um dem Nutzer möglichst weit entgegen zu kommen, können Graphen auch über eine textbasierte Notation (Karteireiter „GraphScript“ oben in Abbildung 5) erstellt werden.

Abbildung 6 zeigt den Assistent zur Erzeugung der Animation des Graphen sowie von Graphenalgorithmen. Hier kann der Benutzer die einzelnen Komponenten des Graphen an von ihm eingegebenen Koordinaten ausrichten. Dazu zählt sowohl der Graph an sich als auch die Adjazenzmatrix, die auch aus der Darstellung entfallen kann. Zu den Animationseigenschaften der Adjazenzmatrix zählen dabei die Schriftgröße sowie die Farben für den Hintergrund, das Gitter, die Kennzeichnung der Elemente und der Gewichte. Grüne Punkte in der Abbildung zeigen an, dass die jeweilige Komponente erfolgreich erstellt wurde. Rote Punkte deuten auf noch nicht erstellte Komponenten, blaue auf eventuell veraltete Daten hin. Auch für den gewählten Algorithmus sind einige Werte einstellbar – neben Farben und Ausrichtung gehört hierzu insbesondere eine Codeanzeige und algorithmenspezifische Parameter („Erweiterte Einstellungen“).

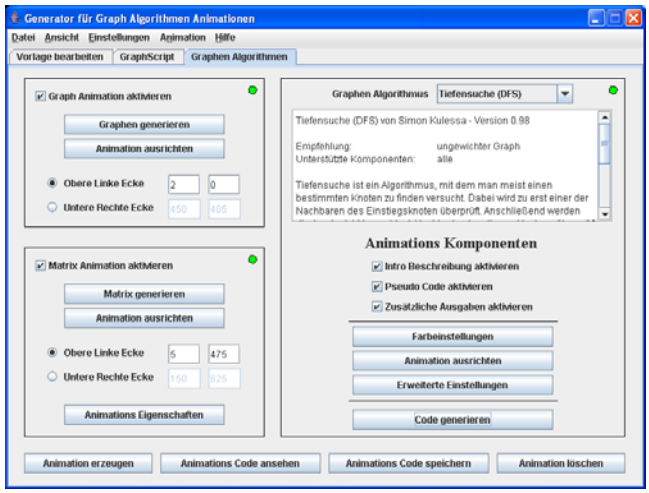

Abbildung 6: Erstellung von Animationen für Graphen und Graphalgorithmen

Abbildung 7 zeigt zwei verkleinerte Fassungen des Ausgabefensters, das die in den Abbildungen 2-4 sowie in Abbildung 5 erstellten Animationen anzeigt.

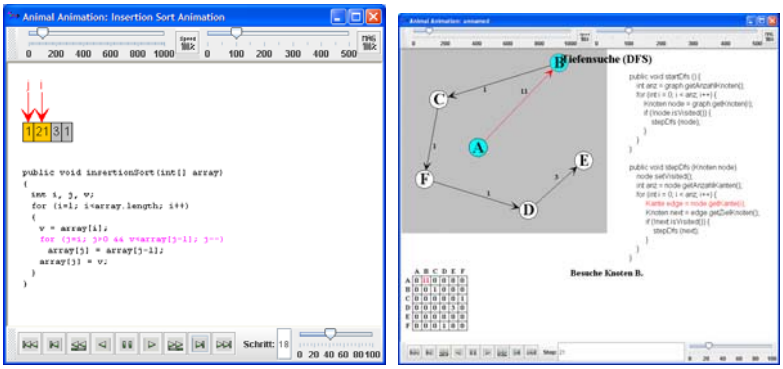

Abbildung 7: Anzeige der generierten Animationsinhalte (links: Insertion Sort, rechts: Tiefensuche im Graph)

4 Das Generator-Framework aus Entwicklersicht

Das Framework ist darauf ausgelegt, die Integration neuer Klassen („Generatoren“) möglichst einfach zu gestalten. So muss eine neue Klasse lediglich ein Interface mit sechs Methoden implementieren, von denen wiederum fünf lediglich für Metadaten zuständig sind: Name, Beschreibung und Beispielcode, wie in Abbildung 3 zu sehen, sowie die Angabe des Generatortyps für die Einsortierung in die Kategorieliste aus Abbildung 2. Die fünfte Methode spezifiziert die Dateiendung des generierten Inhalts für den Speicherungsfilter, wie ebenfalls in Abbildung 3 unten zu sehen ist.

Die Methode für die eigentliche Generierung des Inhalts erhält als Parameter zwei Container: einen für die Eigenschaften der Objekte sowie einen mit den vom Nutzer eingegebenen primitiven Objekten, etwa einem *int*-Feld für einen Sortieralgorithmus.

Die Festlegung der einstellbaren Primitiven und Objekteigenschaften erfolgt in XML. Um den Autor von Mehrarbeit zu entlasten, wird die XML-Datei grafisch erstellt, wie in Abbildung 8 für die englische Sprachversion illustriert. Der Nutzer hat eine Primitive „*array*“ definiert sowie die Eigenschaften „*array*“ (für die Anzeigeeigenschaften eines Feldes), „*iMarker*“ und „*jMarker*“ für die Anzeigeoptionen der Laufvariablen *i*, *j* des Algorithmus für das in Abbildung 7 gezeigte Feld. Das selektierte Element gibt die Farbe der Elemente des Feldes vor und ist hier auf den Wert grün voreingestellt.

Durch ein Häkchen kann markiert werden, ob der Endnutzer eine gegebene Eigenschaft ändern kann oder nicht. So sind die Eigenschaften *fillColor*, *filled*, *depth*, *cascaded* und *vertical* des Elements *array* sowie die Eigenschaft *depth* des Elements *iMarker* als „nicht editierbar“ markiert. Die vom Endnutzer nicht änderbaren Eigenschaften werden im Generator nicht angezeigt (siehe Abbildung 4).

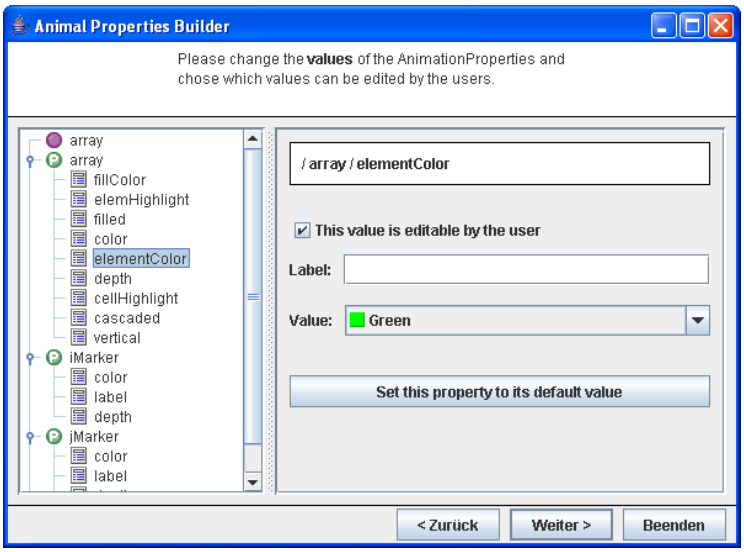

Abbildung 8: Zusammenstellung der editierbaren Primitiven und Eigenschaften

Momentan dient das Framework ausschließlich der Erstellung von AV-Inhalten für das ANIMAL-System, erkennbar an der Dateierweiterung „asu“ in Abbildung 3. Prinzipiell lässt sich das Framework aber auch einfach in anderen Kontexten einsetzen. Eine nahe liegende Anwendung ist etwa der Einsatz als grafisches Konfigurationswerkzeug für Programmieranfänger. So können elegant *int*-Felder direkt an Algorithmen übergeben werden, ohne dass die Studierenden die Arrays im Code fest vorgeben oder sich andererseits mit der aufwändigen Verarbeitung der Kommandozeilenparameter von Java auseinandersetzen müssen.

5 Zusammenfassung und Ausblick

Das in diesem Beitrag präsentierte Framework erleichtert die Erstellung von Inhalten durch eine nutzerfreundliche GUI. Die Komponente gliedert sich in einen Entwickler- und einen Endnutzerteil. Im Entwicklerteil erstellt der Autor eines Generators grafisch eine Spezifikation der relevanten Objekte und Eigenschaften. Diese Spezifikation wird anschließend als XML-Datei abgespeichert. Eine direkte Erzeugung der XML-Inhalte ist natürlich ebenfalls möglich, ist aber sowohl zeitaufwändiger als auch fehleranfälliger als die grafische Erstellung.

Im Endnutzerteil können Inhalte mit wenigen Mausklicken erzeugt werden, indem der Nutzer die Inhaltskategorie und dann den konkreten Inhalt auswählt. Anschließend kann der Nutzer die gewünschten Einstellungen treffen und so etwa das zu sortierende Feld und seine Farbeigenschaften einstellen. Nach der Eingabe des Datennamens werden anschließend die Inhalte automatisch erzeugt und in die vom Nutzer eingegebene Datei gespeichert.

Die Graphenkomponente ist noch nicht in das Framework integriert, erlaubt aber eine ebenso einfache und dennoch komfortable Erstellung von Inhalten. Auch hier kann der Nutzer sowohl die Werte als auch deren Erscheinungsbild einfach und dennoch effektiv an seine Wünsche anpassen.

Sowohl das Generator-Framework als auch die Graphenkomponente sind sehr einfach und schnell zu nutzen. Damit eignen sie sich für erfahrene Lehrkräfte auf der Suche nach einem passenden Beispiel für die Lehrveranstaltung – das sogar „live“ während der Lehrveranstaltung erstellt werden kann – wie für Studierende, die nach weiteren Beispielen zum besseren Verständnis des Stoffes suchen.

Die Erweiterung des Frameworks um neue Generatoren ist ebenfalls sehr einfach, da außer fünf weitgehend trivialen Methoden lediglich eine Methode zu implementieren ist. Diese erhält die vom Nutzer eingestellten Werte und Eigenschaften. Typischerweise wird man daher oft lediglich eine „Wrapper-Klasse“ schreiben, die diese Methode implementiert und die übergebenen Parameter „passend“ in die eigentliche Klasse für die Inhaltserstellung weiterleitet.

Für die Weiterentwicklung des Systems wollen wir zusätzliche Objekttypen mit ihren Eigenschaften hinzufügen. So können Aufzählungstypen etwa bei Sortierverfahren die Auswahl aus einer vorgefertigten Anzahl voreingestellter Werte oder Wertanordnungen erlauben, wie „aufsteigend sortiert“, „absteigend sortiert“, „überwiegend sortiert“ und „zufällig“. Auch die Graphkomponente soll in das Framework integriert werden.

Eine geplante formale Evaluation der Nutzung, sowohl aus Autoren- als auch als Nutzersicht, soll den Wert des Frameworks absichern. Zusätzlich ist zu untersuchen, inwieweit das System auch weniger technikaffine Nutzer (vor allem außerhalb der Informatik) unterstützt. Dabei sind Parameter wie didaktischer Wert, Lerneffizienz, Nutzbarkeit und Skalierbarkeit für neue Algorithmen ebenso von Interesse wie der Umfang der mit dem System interaktiv bearbeitbaren Aufgaben.

Schließlich prüfen wir die Umsetzbarkeit eines „Batch Modes“. Er erlaubt die Erstellung von Inhalten ohne Nutzerinteraktion, insbesondere auch ohne grafische Schnittstelle. Bei entsprechender Struktur könnte der Nutzer so viele Inhalte „auf einen Rutsch“ erstellen lassen, indem er das Framework durch eine Schleife mit einer entsprechenden Anzahl an Konfigurationsdaten füttert. So sind etwa leicht Skripte denkbar, die alle für eine gegebene Vorlesung nötigen Animationen auf einen Schlag erzeugen.

Literaturverzeichnis

[Bl06] Blumberg, M. et al.: j-Algo, The Algorithm Visualization Tool. Online: http://j-algo.binaervarianz.de, gesehen am 10. 3. 2006.

[JFH00] Jarc, D. J.; Feldman, M. B.; Heller, R. S.: Assessing the benefits of interactive prediction using Web-based algorithm animation courseware. In: Proc. 31st SIGCSE Technical Symposium on Computer Science Education, Austin, TX. ACM Press, New York, 2000; S. 377-381.

[Ko03] Korhonen, A.: Visual Algorithm Simulation. Doktorarbeit, Fachbereich Informatik der Helsinki University of Technology, 2003.

[RH04] Rößling, G.; Häußge, G.: Towards Tool-Independent Interaction Support. In (Korhonen, A., Hrsg.): Proceedings of the Third International Program Visualization Workshop, University of Warwick, England. University of Warwick Press, Coventry, UK, 2004; S. 110-117.

[Ro02] Rodger, S. H.: Introducing Computer Science Through Animation and Virtual Worlds. In: Proc. 33rd SIGCSE Technical Symposium on Computer Science Education, Northern Kentucky. ACM Press, New York, 2002; S. 186-190.

[Rö04] Rößling, G.: Integration von Algorithmenanimationen in die Lehre mittels ANIMAL. In: (Dadam, P.; Reichert, M., Hrsg.): Tagungsband 1 zur Informatik 2004, Jahrestagung der Gesellschaft für Informatik, Ulm. Gesellschaft für Informatik e.V., Bonn, 2004; S. 404-408.

[Na02] Naps, T. L. et al.: Exploring the Role of Visualization and Engagement in Computer Science Education. In: inroads - Paving the Way Towards Excellence in Computing Education, Vol. 35, Nr. 2. ACM Press, New York, 2003; S. 131-152.