# Das Virtuelle Informatiklabor
## Konzeption, Design und Realisierung einer E-Learning-Umgebung zum Erlernen von Algorithmen

Claudia Schmidt, Volker Sänger
Fakultät Medien und Informationswesen
Hochschule Offenburg
Badstr. 24, 77652 Offenburg
c.schmidt | volker.saenger AT hs-offenburg.de

**Zusammenfassung**

Das Virtuelle Informatiklabor soll Schülern und Studierenden den übergroßen Respekt vor dem Fach Informatik nehmen und sie beim Lernen der Inhalte unterstützen. Zu diesem Zweck werden grundlegende Algorithmen der Informatik anhand konkreter Aufgabenstellungen in interaktiven Anwendungen behandelt, um den Lernenden das explorative Erkunden zu ermöglichen. Animationen sollen das Verstehen fördern, Experimente das eigenständige, durch vielfältige Hilfen unterstützte Anwenden und Umsetzen des Gelernten. Der erste Themenbereich im Virtuellen Informatiklabor umfasst die Rekursion, die in mehreren Anwendungen präsentiert wird.

**Abstract**

The Virtual Computer Science Lab aims to decrease the great respect that pupils and students have for the subject of Computer Science and tries to support their learning. For this purpose fundamental algorithms of Computer Science are handled with the help of concrete tasks in interactive applications in order to allow learners to explore those algorithms. Animations promote the understanding, experiments foster independent, by a variety of aids supported employment of the lessons learned. The first topic in the Virtual Computer Science Lab comprises recursion, which is presented in several applications.

## 1 Das Virtuelle Informatiklabor - Idee und Zielsetzung

In seinem *konzeptblog* (Wedekind, 2012) fragt Joachim Wedekind, ob „alle Programmieren lernen sollten“. Und er erwähnt Douglas Rushkoff, der bereits im Titel seines Buches „Program or be Programmed“ (Rushkoff, 2011) seine persönliche Antwort darauf gibt, indem er Programmieren als eine Art Kulturtechnik des digitalen Zeitalters bezeichnet. Das Erlernen von Informatikinhalten ist aber alles andere als eine Selbstverständlichkeit; dies erfahren die Autoren des vorliegenden Artikels ständig. Seit über 10 Jahren lehren sie Informatikthemen in der Fakultät Medien und Informationswesen der Hochschule Offenburg. Immer wieder machen Studierende deutlich, dass Informatik "schwierig" und nur durch "mühsames Büffeln zu erlernen ist" und deswegen in dem interdisziplinären Kontext der Fakultät unattraktiver als die alternativen Fachrichtungen erscheint. Viele Studierende haben (über-)großen Respekt vor dem Fach und trauen sich nicht an die Inhalte und Aufgaben heran.

Immer wieder zeigt sich aber auch, dass Erfolge der Studierenden in Prüfungen, Projekten und Praktika zu einem Umdenken führen: Informatik erscheint doch machbar, kann interessant sein und sogar Spaß machen. Leider finden noch zu wenige Studierende diesen positiven Zugang zur Informatik, obwohl seit Jahren bekannt ist, dass es zu wenige Informatik-Absolventen gibt (Hanselmann, 2009) und deshalb die Berufschancen für Absolventen exzellent sind (Weidner, 2011). Diese Sachlage veranlasste die beiden Autoren zu Überlegungen, wie bei Studierenden und Schülern das Interesse für Informatik geweckt und ihnen der übergroße Respekt vor dem teilweise nur vermeintlich schwierigen Fach genommen werden kann.

In den letzten Jahren entwickelten die Autoren zusammen mit Studierenden mehrere E-Learning-Umgebungen, die parallel zu den Vorlesungen die komplexen Sachverhalte visualisieren und über interaktive Applikationen das selbständige Lernen an konkreten Beispielszenarien ermöglichen (Sänger & Schmidt 2007; Schmidt, Sänger & Endres 2009, Sänger & Schmidt 2010; Pfannstiel, Sänger & Schmidt, 2009). Basierend auf diesen Erfahrungen, entstand die Idee, eine interaktive webbasierte E-Learning-Umgebung zu entwickeln, mit deren Hilfe Schüler und Studierende sich der Informatik annähern, mit der sie die Informatik entdecken können. In dieser Umgebung, genannt das **Virtuelle Informatiklabor**, werden grundlegende Algorithmen der Informatik behandelt, weil sie für das Fach zentral sind, weil sie die erforderliche Denk- und Herangehensweise an das Fach nach Meinung der Autoren am besten repräsentieren.

Als Anker sollen praktische Probleme des Alltags dienen, die den Lernenden bekannt sind und ihr Interesse wecken. Die Anwendungsorientierung und der Problemlösungsaspekt und damit die Relevanz der Informatik für unser tägliches Leben sollen im Vordergrund stehen. Fragen wie "Wie funktioniert ein Routenplaner?" (Gallenbacher, 2008) oder "Wie löse ich das Spiel Türme von Hanoi?" führen direkt zu Konzepten und Algorithmen der Informatik. Wichtige Fragestellungen, die in diesem Kontext auftreten und im Rahmen der Entwicklung zu beachten sind, befassen sich mit den Auswirkungen von Algorithmenvisualisierungen auf den Lernerfolg. Welche Elemente sollten interaktive E-Learning-Anwendungen in der Informatik besitzen, um die Lern- und Behaltensleistung nachhaltig zu verbessern? Wie ist es möglich, Algorithmen explorativ zu erlernen? Neben den Aspekten des Lernens spielte auch die Zielgruppe eine wichtige Rolle. Es sollten grafisch ansprechende, interaktive Anwendungen entwickelt werden, um Schüler und junge Studierende direkt anzusprechen und zu motivieren, sich mit der Thematik zu befassen.

Der Artikel ist wie folgt aufgebaut. In Kapitel 2 werden die wichtigen Grundlagen für die Konzeption des Virtuellen Informatiklabors vorgestellt. Kapitel 3 beschreibt dessen grundlegende Entwurfsentscheidungen. Darauf aufbauend wird in Kapitel 4 die Umsetzung der Entwurfsentscheidung innerhalb der Umgebung erläutert. Kapitel 5 enthält die Beschreibung des ersten Themenbereichs, der Rekursion, bevor der Artikel mit einer Zusammenfassung schließt.

# 2 Grundlagen für das E-Learning von Algorithmen

## 2.1 Taxonomien von Lehr- und Lernzielen

Lehr- und Lernziele beschreiben die Fähigkeiten, die ein Lernender nach erfolgreichem Absolvieren einer Lerneinheit erworben haben soll. In der Vergangenheit wurden mehrere Taxonomien entwickelt, die einzelne Zustände im Lernprozess beschreiben und klassifizieren. Weit verbreitet ist die Taxonomie von Bloom und den darauf aufbauenden Arbeiten von Anderson & Krathwohl für kognitive Lernziele (Anderson & Krathwohl, 2001). Diese Taxonomien sind unabhängig vom jeweiligen Fach, und daher argumentieren Fuller et al. (Fuller, Johnson, Ahoniemi, Cukierman, Hernán-Losada, Jackova, Lahtinen, Lewis, McGee Thompson, Riedesel, Thompson, 2007), dass für die Informatik einige Besonderheiten gelten, die nach einer eigenen speziellen Taxonomie verlangen. Sie gehen davon aus, dass die Informatik eine anwendungsorientierte Disziplin ist, bei der es besonders wichtig ist, das Wissen anwenden zu können und damit Probleme zu lösen. Daraus leiten sie basierend auf der Bloomschen Taxonomie eine neue zweidimensionale Taxonomie für die Informatik und speziell für das Programmierenlernen ab (siehe Tabelle 1).

Wichtig für diese neue Taxonomie ist die Erkenntnis, dass das Verstehen von Programmcode und das Programmieren selbst zwei komplett unterschiedliche Fähigkeiten sind. Dies führt zu einer zweidimensionalen Anordnung der Lernziele, bei der horizontal die verschiedenen Level zur Interpretation von Programmcode (Interpreting) und vertikal die Level zum Programmieren (Producing) angeordnet sind. Dabei sind die Level jeder Ebene gestuft, das bedeutet für die Ebene Producing, dass auf der Stufe Apply einzelne Programmierkonstrukte angewendet werden können und darüber auf der Stufe Create dann komplette Programme geschrieben werden können.

| | | | | | |
|---|---|---|---|---|---|
| **Producing** | Create | | | | |
| | Apply | | | | |
| | None | | | | |
| | | Remember | Understand | Analyze | Evaluate |
| | | **Interpreting** | | | |

Tabelle 1: Graphische Darstellung der zweidimensionalen Lernziel-Taxonomie (Fuller et al., 2007)

## 2.2 Lernen und Visualisieren von Algorithmen und Programmen

Das Erlernen von Algorithmen und auch das Umsetzen in Programme stellen hohe kognitive Anforderungen an die Lernenden. Dies ist auch daran zu erkennen, dass viele Studienanfänger der Informatik ihr Studium abbrechen. Die Drop-Out-Quote des Informatikstudiums in Deutschland liegt mit über 30 % sehr hoch (Heublein, Hutzsch, Schreiber, Sommer & Besuch, 2009), aber auch im internationalen Trend.

Ben-Ari untersuchte die Ausbildung der Informatik unter konstruktivistischen Aspekten (Ben-Ari, 1998). Basierend auf der konstruktivistischen Lerntheorie konstruieren die Lernenden aktiv neues Wissen, indem sie Beobachtungen und Wahrnehmungen überdenken und mit ihrem bisherigen Vorwissen verknüpfen. Lernende absorbieren also nicht passiv die vorgestellten Konzepte, sondern jeder baut seine eigenen Modelle auf. Ben-Ari argumentiert nun, dass fehlendes Vorwissen und damit fehlende Modelle beim Erlernen der Informatik sehr schnell zu Frustration führen, so dass der Anknüpfpunkt fehlt und damit alles komplett neu erlernt werden muss. Er plädiert dafür, dass die Modelle und Algorithmen der Informatik explizit gelehrt werden und nicht einer zufälligen und falschen Konstruktion überlassen werden dürfen. Eine Möglichkeit, Algorithmen zu lehren, besteht darin, diese zu visualisieren (Ben-Ari, 2002).

Die Bildungsstandards Informatik der Gesellschaft für Informatik e.V. fordern, dass Schülerinnen und Schüler aller Jahrgangsstufen Algorithmen aus verschiedenen Anwendungsbereichen kennen sollen, und sie sollen in der Lage sein, Algorithmen zu lesen und zu interpretieren. Hilfsmittel sind dabei formale Darstellungsformen von Algorithmen, wie etwa Programmablaufpläne oder Struktogramme (Brinda, Fothe, Friedrich, Koerber, Puhlmann, Röhner & Schulte, 2008, Seite 30/31).

Animationen werden bereits verbreitet zum Erlernen von Algorithmen und Datenstrukturen eingesetzt, da sie nicht nur eine graphische Darstellung erlauben, sondern auch den Ablauf mit einzelnen Zuständen und die Übergänge zwischen diesen Zuständen visualisieren. Eine Studie mit über 500 Algorithmen-Visualisierungen (Shaffer, Cooper, Alon, Akbar, Stewart, Ponce & Edwards, 2010) zeigt, dass sowohl Lehrende als auch Lernende gerne mit Visualisierungen arbeiten, sich aber die Frage stellt, welche Auswirkungen der Einsatz von Visualisierungen auf den Lernerfolg besitzt. Hier existieren unterschiedliche Untersuchungsergebnisse, die von keinen Verbesserungen durch Visualisierungen bis zu einem besseren Verständnis reichen. Daher stellt sich die Frage, welche Randbedingungen einen höheren Lernerfolg mit Algorithmen-Visualisierungen ermöglichen.

Hundhausen erkannte schon 1998: "The most significant factor (in learning) appears to be not what AV (algorithm visualization) viewers see, but what they do (with the visualizations)" (Hundhausen, 1998). In einer weiteren Studie kommt er zu dem Ergebnis: "the most successful educational uses of AV technology are those in which the technology is used as a vehicle for actively engaging students in the process of learning algorithms." (Hundhausen, Douglas & Stasko, 2002). Diverse weitere Studien zeigen ebenfalls, dass Lernende Algorithmen effizient erst durch einen aktiven Umgang mit dem Algorithmus selbst erlernen.

Im Vergleich mit der Visualisierung von Algorithmen existieren deutlich weniger Tools zur Visualisierung von Programmen. Der Grund liegt darin, dass ein Tool zur **Visualisierung von Programmen** einen relativ hohen Entwicklungsaufwand erfordert. Ein solches Tool, das zum Programmierenlernen eingesetzt wird, muss ein entwickeltes Programm parsen, analysieren und interpretieren können. Die Programmausführung kann schrittweise mit dem Zustand der Variablen visualisiert werden.

Bestehende Tools zur Visualisierung von Programmen zeigen parallel den Programmcode und während des Ablaufs die Werte der Variablen und markieren dabei den aktuellen Ausführungsschritt. Sie unterscheiden sich

in der unterstützten Programmiersprache, dem Detaillierungsgrad der Anzeige, der Steuerung und der Interaktion mit dem Lernenden (Helminen & Malmi, 2010). Die meisten Tools basieren dabei auf Java und bieten kaum Möglichkeiten zu aktiven Interaktionen.

## 2.3 Interaktivität in Lernprogrammen

Lernende lösen in Lernanwendungen Aufgaben oder Probleme, indem sie mit der jeweiligen multimedialen Anwendung interagieren. Die **Interaktivität** besitzt dabei einen großen Einfluss auf die kognitiven und die motivationalen Prozesse, die beim Lernen ablaufen. Erst durch die Interaktivität wird ein exploratives Lernen möglich und ein aktives Denken und selbständiges Entscheiden gefördert (Strzebkowski & Kleeberg, 2002).

In der Literatur findet sich eine Klassifizierung von Interaktionen in zwei unterschiedliche Formen, wobei die Grenze zwischen den beiden Formen fließend ist (Strzebkowski & Kleeberg, 2002):

- **Steuerungsinteraktionen** umfassen Navigations- und Systemfunktionen, wie beispielsweise die Auswahl bestimmter Inhalte, die Steuerung und Wiedergabe von zeitbasierten Lernanwendungen, die Wahl eines eigenen Lernweges oder eine Einzelworteingabe.
- **Didaktische Interaktionen** unterstützen direkt den Erkenntnisprozess und werden ermöglicht durch exploratives Lernen, bei dem der Lernende Bedeutungen selbst erschließt oder zu neuen Einsichten gelangt. Sie umfassen die Steuerung von interaktiven Anwendungen, die Eingabe von Antworten auf komplexe Fragestellungen, die Modifikation oder Anpassung von Daten und Lernwegen, das Erzeugen neuer multimedialer Daten oder Objekte und ein adaptives tutorielles Feedback oder eine adaptive Hilfe vom Lernprogramm.

Die meisten Lernprogramme bieten als Interaktionsformen nur einfache Steuerungsinteraktionen an. Die Lernenden bleiben in der Rolle eines passiven Rezipienten, da sie nur Objekte oder Inhalte bestimmen und auswählen können (Strzebkowski & Kleeberg, 2002). Didaktische Interaktionen setzen erst dann ein, wenn ein aktives Denken und Informationsverarbeitungsprozesse angestoßen werden. Möglich ist dies bei einem explorativen Lernen, bei dem der Lernende Bedeutungen selbst erschließt oder zu neuen Einsichten gelangt.

# 3 Entwurf des Virtuellen Informatiklabors

Das Virtuelle Informatiklabor setzt die im vorigen Kapitel beschriebenen Grundlagen für das E-Learning von Algorithmen um. Es unterstützt sowohl die Dimension des Verstehens (Interpreting) als auch die des Anwendens (Producing) gemäß Tabelle **1**. Die Algorithmen werden so visualisiert, dass dem Lernenden vielfältige Interaktionsmöglichkeiten geboten werden, um das aktive Denken zu fördern.

## 3.1 Struktur des Labors

Das Virtuelle Informatiklabor wurde als offener Interaktionsraum in Form eines Hypermedia-Systems konzipiert, um damit eine Zielgruppe zu erreichen, die über unterschiedliches Interesse und Vorwissen verfügt und unterschiedliche Lerngewohnheiten besitzt. Dabei können jedoch Lernprobleme durch die Nutzung von Hypermedia entstehen, wie Desorientierung und kognitive Überlast (Tergan, 2002), denen jedoch über geeignete Navigations- und Orientierungshilfen (Haack, 2002) entgegengewirkt wird.

Im Virtuellen Informatiklabor sollen unterschiedliche Themen der Informatik explorativ erlernt werden. Jeder Themenkomplex beginnt mit einer kurzen und prägnanten Beschreibung des Themas (z.B. Rekursion, Sortieren). Diese **Einstiegsseite** dient zum einen dazu, den Themenkomplex inhaltlich kurz vorzustellen, zum anderen aber auch zur Motivation, um sich mit den Algorithmen zu diesem Thema zu beschäftigen. Grundsätzlich soll das Virtuelle Informatiklabor multiple Kontexte und Perspektiven unterstützen, und daher werden pro Themenblock mehrere Algorithmen zum Erlernen des Themas eingesetzt. Dies führt dazu, dass in einer Ebene unter der thematischen Einstiegsseite zunächst auf einer **Algorithmenseite** die einzelnen Algorithmen kurz vorgestellt

werden und dann in der nächst tieferen Ebene die einzelnen Lernanwendungen zum jeweiligen Algorithmus zu finden sind.

Zu jedem Algorithmus existieren mehrere Anwendungsbeispiele, die sich grundsätzlich in zwei unterschiedliche Gruppen einteilen lassen (vergleiche Tabelle **2**). **Anwendungen mit Instruktionscharakter** dienen dem Erlernen eines Algorithmus. Dies erfolgt an ausgewählten Beispielen, die über ein Kontrollpanel gesteuert werden können. Ein aktiver Umgang mit dem Algorithmus und damit ein effizienteres Lernen (Hundhausen, 1998) wird erreicht, indem in der Regel die Eingabedaten vom Lernenden gewählt werden können oder aber Fragen zum Algorithmus gestellt werden (Naps, 2005).

| | **Anwendungen mit Instruktionscharakter** | **Anwendungen mit Konstruktionscharakter** | |
|---|---|---|---|
| | | **Level 1** | **Level 2** |
| **Ziel** | Erlernen des Algorithmus | Algorithmus auf neue Problemstellungen anwenden können | Algorithmus selbst entwickeln können |
| **Realisierung** | über interaktive Animationen | Experimente (an einem konkreten Beispiel) | mit einer Toolbox den Algorithmus als Struktogramm entwickeln |
| **Interaktions-elemente** | Kontrollpanel zur Steuerung, Auswahl von Eingabe- oder Steuerungsparametern | direktes Interagieren am Anwendungsbeispiel | Elemente der Toolbox, Kontrollpanel zur Steuerung |
| **Lernziele** | Interpreting (Remember, Understand, Analyze) | Apply | Create |

Tabelle 2: Unterscheidung von Anwendungen zur Instruktion und Konstruktion

Ist ein Algorithmus bereits bekannt, so können Anwendungen bearbeitet werden, die **Konstruktionscharakter** besitzen. Dazu zählen Experimente an einem konkreten Beispiel und die Entwicklung von eigenen Algorithmen und Programmen. Das Ziel liegt hier im Transfer des Erlernten – dies soll in neuen Szenarien angewendet werden oder genutzt werden, um eigenständig einen Algorithmus für ein spezielles Problem zu entwickeln. Diese Anwendungen verlangen ein hohes Maß an Interaktion und besitzen keinen vorgegebenen Lösungsweg. Lernende haben die Chance, auch Fehler zu machen und aus diesen Fehlern zu lernen.

Die Darstellung der Algorithmen soll in der ersten Version des Virtuellen Informatiklabors ausschließlich auf **Struktogrammen** basieren, da diese durch die graphische Darstellung leicht lesbar sind und das Verständnis sowie eine strukturierte Entwicklung von Algorithmen unterstützen. Diese Entscheidungen sind von dem Ziel geleitet, zunächst ein grundlegendes Verständnis von Algorithmen zu erreichen, bevor eine Programmierung angegangen wird. Um die Komplexität zu reduzieren und das Verständnis zu erleichtern, wird der Fokus auf die algorithmischen Grundbausteine Sequenz, Auswahl und Wiederholung gelegt (Brinda, Fothe, Friedrich, Koerber, Puhlmann, Röhner & Schulte, 2008, Seite 30).

Mit dieser Vorgehensweise befindet sich das Virtuelle Informatiklabor auf der Ebene der visuellen Algorithmenentwicklung, also auf einer höheren Lernebene als die Visualisierung von Algorithmen (Interpreting), gleichauf mit der visuellen Programmierung (Create). Allerdings werden durch die höhere Abstraktionsebene der Struktogramme die typischen Probleme der Programmierung, wie Syntaxfehler und deren zeitaufwändige Suche, reduziert, so dass sich der Lernende mehr auf den eigentlichen Algorithmus konzentrieren kann.

Für eine spätere Erweiterung, um Programmieren zu lernen, ist auch der Einsatz von Programmcode möglich. Sowohl in den Instruktions- als auch in den Konstruktionsanwendungen wird, falls möglich, parallel eine Darstellung als Struktogramm und als Anwendungsbeispiel angestrebt. Damit lässt sich ähnlich zu den

vorgestellten Programmvisualisierungen (vergleiche Abschnitt 2.2) ein Algorithmus schrittweise am Beispiel und im Struktogramm nachverfolgen.

Entsprechend der in Abschnitt 2.1 vorgestellten **Lernziele**, dienen alle Anwendungen mit Instruktionscharakter dem Verstehen von Algorithmen (entsprechend der horizontalen Achse Interpreting), während die Anwendungen mit Konstruktionscharakter die Entwicklung von Algorithmen zum Ziel haben und somit unter die Level Apply und Create auf der Achse Producing fallen. Grundsätzlich liegen die Lernziele des Virtuellen Informatiklabors auf der Achse Interpreting hauptsächlich in den ersten drei Stufen: die Algorithmen sollen erinnert werden (Stufe 1: Remember), verstanden werden (Stufe 2: Understand) und analysiert werden können (Stufe 3: Anaylze). Dabei sind über die Konstruktionsanwendungen alle vertikalen Stufen auf der Achse Producing oberhalb der Felder Remember, Understand und Analyze erreichbar.

## 3.2 Entwurfsentscheidungen der Lernanwendungen

Beim Entwurf aller Anwendungen, sowohl der Animationen zum Erlernen der Algorithmen als auch der Experimente und dem Entwickeln und Programmieren von Algorithmen, stehen die **Interaktionen** des Lernenden mit der Anwendung im Mittelpunkt. Durch die Interaktion soll der Lernerfolg erhöht werden (vergleiche auch Abschnitt 2.2). Alle Animationen mit Instruktionscharakter sehen als Interaktionen entweder die Wahl von Eingangsdaten oder die Beantwortung von Fragen im Ablauf vor. Bei Experimenten und dem Entwickeln von Algorithmen wird eine noch größere Interaktion gefordert und damit ein selbständiges Konstruieren von Lösungswegen erlaubt.

Weitere Entscheidungen, die für alle Anwendungen getroffen wurden, sind:

- **Angabe der Aufgabenstellung**: in jeder Anwendung wird in einer kurzen und prägnanten Aufgabenbeschreibung dargelegt, was die eigentliche Aufgabe ist und wie die Interaktion mit der Anwendung erfolgt.
- **Direkter Zugriff auf Hintergrundinformation**: über Tooltips, ein Lexikon und ein Glossar hat der Lernende jederzeit Zugriff auf zusätzliche Informationen, die bei der Bearbeitung der Aufgabe unterstützen (vergleiche Abschnitt 4.3).
- **Unterstützung durch eine kontextsensitive Hilfe**: Benötigt der Lernende Unterstützung beim Lösen der Aufgabe, so ist aus jeder Anwendung heraus direkt eine kontextsensitive Hilfe anwählbar (vergleiche Abschnitt 4.3).
- **Verständliche Beispiele, ausgewählte Eingangsdaten und Beschränkung auf wesentliche Programmierelemente**: alle Anwendungen sind "didaktisch bereinigt". Nur die zum Erlernen des Algorithmus notwendigen Informationen werden in den Anwendungen eingesetzt. Damit kann der Algorithmus gut gelernt werden, ohne den Lernenden abzulenken, zu langweilen oder zu verwirren.

### 3.2.1 Die Instruktionsanwendungen – interaktive Animationen

Ziel der Instruktionsanwendungen ist das Erlernen von Algorithmen. Dazu werden interaktive Animationen eingesetzt, die vom Lernenden aktiv gesteuert werden. Ein Schwerpunkt beim Entwurf dieser Anwendungen wurde auf die Interaktion der Lernenden mit der Lernanwendung gelegt. Es soll nicht nur ein "Viewing", ein Betrachten von Abläufen stattfinden, sondern mit der Anwendung interagiert werden, um damit den Lernerfolg zu erhöhen.

Für die interaktiven Animationen wurden die folgenden Entwurfsentscheidungen getroffen:

- **Wahl der Eingabedaten**: Bei den interaktiven Animationen sind unterschiedliche Eingabedaten wählbar. Damit können die Lernenden einerseits Hypothesen aufstellen und diese anschließend mit Hilfe der Animation überprüfen. Andererseits können über spezielle Eingabedaten besondere Verhaltensweisen eines Algorithmus vorgestellt werden. Lernende können hiermit Sonderfälle von Algorithmen kennenlernen (Faltin, 1999). Die Wahl der Eingabedaten (Changing nach Naps (2005)) unterstützt einen aktiven Umgang mit den Algorithmen und soll damit die Lerneffizienz erhöhen. Um

die Lernenden nicht zu überfordern, wird in der Regel eine Vorauswahl an Eingabedaten angeboten, aus der Lernende ihre Eingaben wählen können.

- **Erklärungen der Schritte:** Können für einzelne Schritte Zusatzinformationen das Verständnis erleichtern, so werden diese in einem separaten Bereich angezeigt. Darüber hinaus werden optionale Erklärungen zu Struktogrammen oder Pseudocode über eine Tooltip-Funktion angeboten. Hat der Lernende Bedarf an einer Erklärung, so bewegt er einfach die Maus über das gewünschte Element und erhält direkt die entsprechende Information.
- **Anzeige des Algorithmus**: Nach Möglichkeit wird der Algorithmus als Struktogramm oder Pseudocode angezeigt und dabei wird Schritt-für-Schritt der Ablauf korrespondierend zwischen einem angezeigten Beispiel und dem Algorithmus dargestellt. Damit kann ein Lernender direkt die Auswirkungen des Algorithmus am Beispiel verfolgen.
- **Flexible Ablaufsteuerung**: Jede Animation ist über ein Kontrollpanel steuerbar mit den Elementen Play/Pause, Einzelschritte vor und zurück, Sprung an Anfang und Ende. Über Play kann die gesamte Animation abgespielt werden, aber zum Lernen ist es aus didaktischer Sicht wichtig, dass einzelne Schritte vom Lernenden ausgewählt und konkret über die Einzelschritte nachvollzogen werden können. Über schrittweises Zurückgehen kann bei Verständnisproblemen an dem Punkt angeknüpft werden, der noch verstanden wurde.
- **Interaktive Voraussagen:** Bei komplexeren Algorithmen wird der Lernende an wichtigen Punkten mit Fragen zum Nachdenken angeregt. So kann er sein bisheriges Verständnis überprüfen, eventuell nochmals zurückgehen und ist bereit für den weiteren Ablauf.
- **Weiche Übergänge** sollen den Lernenden helfen, die Veränderungen von einem Schritt auf den nächsten nachvollziehen zu können, indem Veränderungen fließend erfolgen, nicht schlagartig.

### 3.2.2 Entwurf der Konstruktionsanwendungen

Für den Lernerfolg ist es wichtig, dass die Lernenden sich aktiv mit dem Lerngegenstand auseinandersetzen (Hundhausen, 1998). Daher besteht ein Schwerpunkt des Virtuellen Informatiklabors aus Anwendungen, in denen die Lernenden eigene Lösungen und eigene Algorithmen entwickeln können. Dabei soll nicht nur ein Schritt weitergedacht und dieser direkt überprüft werden, sondern komplette Lösungsideen für konkrete Aufgabenstellungen entwickelt werden. Wichtig ist in diesem Zusammenhang eine gewisse Offenheit der Themenstellung (Arnold, Hartmann & Reichert, 2005), die alternative Lösungsideen zulässt.

Das Ziel aller Konstruktionsanwendungen besteht darin, das in den Instruktionsanwendungen erworbene Wissen auf neue Problembereiche zu transferieren und damit einerseits konkrete Anwendungsbeispiele zu lösen und andererseits selbständig neue Algorithmen zu entwickeln. Grundsätzlich werden im Virtuellen Informatiklabor zwei Arten von Konstruktionsanwendungen unterschieden (vergleiche Tabelle **2**):

- **Experimente** geben ein konkretes Szenario vor, in dem der Algorithmus angewendet werden soll. Ein Beispiel ist das bekannte Knobelspiel "Türme von Hanoi" – bei dem in einer Animation versucht werden soll, eine Lösung für den rekursiven Algorithmus zu finden.
- **Algorithmen entwickeln** geht einen Schritt weiter. Basierend auf dem erworbenen Wissen soll ein Algorithmus für eine angeführte Problemstellung entworfen werden. Die formale Darstellung in Form von Struktogrammen hat den Vorteil, dass die Lernenden sich auf den Algorithmus selbst konzentrieren können und nicht erst auf das Erlernen einer Programmiersprache und sich auch nicht mit Datentypen auseinandersetzen müssen.

In beiden Fällen besteht das Ziel darin, dass Lernende eine Experimentierumgebung vorfinden, in der sie einen großen Spielraum besitzen: Sie dürfen Fehler machen, eigene Lösungsansätze ausprobieren und erleben somit die Suche nach einer Lösung als Herausforderung. Gleichzeitig sind die Lernenden jedoch nicht sich selbst überlassen. Das System erkennt Fehler und weist die Lernenden darauf hin. Über die gestufte Hilfe und das Feedback erhalten sie Hilfestellungen und werden falls nötig zur Lösung hingeführt.

Aus diesen Überlegungen heraus, werden die folgenden Anforderungen an die Experimente des Virtuellen Informatiklabors gestellt:

- Die **Szenarien** zum Experimentieren sollten **anwendungsorientiert** gewählt werden, d.h. dass ein Algorithmus nicht um seiner selbst willen sondern anhand eines möglichst realen Problems oder Einsatzgebietes eingeführt wird, um damit ein anwendungsorientiertes Wissen zu fördern und um die Motivation zum Experimentieren zu steigern.
- Die Lernenden sollten über unterschiedliche Interaktionen (z.B. Drag-and-Drop, Selektieren, Verbinden), das Szenario selbständig verändern können. Dabei wird ein **freies Experimentieren**, soweit dies möglich ist, unterstützt.
- Es werden **unterschiedliche** und damit auch nicht korrekte **Lösungswege** zugelassen. Ist eine Lösung erreicht, so wird diese über das Feedback bewertet.
- Das **Feedback** erfolgt **verzögert**, da dies zum einen das experimentelle Lernen unterstützt und damit aus Fehlern gelernt werden kann. Zum anderen wird die langfristige Behaltens- und Transferleistung über diese Feedbackvariante gefördert.
- Neben dem Szenario zum Experimentieren soll der entsprechende **Algorithmus halbformal oder formal** angezeigt werden. Es muss für den Lernenden möglich sein, den Algorithmus nochmals nachzuvollziehen und parallel auf das Szenario anzuwenden.
- **Unterstützung** erfährt der Lernende über die Hilfe. Hier wird er kontext-sensitiv über Fragestellungen auf die Themen gelenkt, die ihm zur Lösung benötigte Informationen bieten.

Für die Anwendungen zum **Entwickeln von Algorithmen** wurden die folgenden Anforderungen abgeleitet:

- Ausgehend von der Aufgabenstellung soll über eine **Tool-Box** ein Struktogramm für den entsprechenden Algorithmus entwickelt werden. Dabei sind alternative Lösungen möglich – auch Lösungen, die nicht das richtige Ergebnis liefern.
- Bei der Entwicklung des Struktogramm wird der Lernende über ein **gestuftes Feedback** und eine gestufte Hilfe unterstützt und falls nötig schrittweise zur Lösung hingeführt.
- Die **syntaktische und semantische Korrektheit** der Lösung wird überprüft.
- Das entwickelte Struktogramm wird anhand des zugrunde liegenden Anwendungsbeispiels animiert. Dabei können unterschiedliche Lösungen animiert werden, teilweise auch mit Entwurfsfehlern, so dass der Lernende die Auswirkungen eigener Fehler deutlich erkennen und korrigieren kann.

Für die Entwicklung von Algorithmen wurde eine generische **Tool-Box** entwickelt, mit der Struktogramme von den Lernenden erstellt werden können. Diese Tool-Box war bei der Entwicklung und Implementierung relativ komplex, erlaubt aber nun die schnelle Integration ganz unterschiedlicher Algorithmen. Die Tool-Box soll leicht bedienbar, jederzeit änderbar und auf unterschiedliche Algorithmen konfigurierbar sein. Im GUI der Toolbox wird das Struktogramm dargestellt, und über Editorfunktionen können einzelne Struktogrammblöcke hinzugefügt, verändert und gelöscht werden. Dabei steht die Einfachheit im Vordergrund: Es werden nur die wirklich benötigten Struktogrammelemente zur Verfügung gestellt, mit denen Folge, Verzweigung und Wiederholung realisiert werden können. Außerdem soll der Lernende nur mit Variablen und Funktionen arbeiten, die im Algorithmus benötigt werden. Daher ist auch hier nur eine Auswahl aus vordefinierten Variablen und Funktionen möglich. Erstellte Struktogramme können über Dateifunktionen gespeichert und wieder geöffnet werden. So können die Lernenden bereits entwickelte Lösungen wiederverwenden.

Haben Lernende ein Struktogramm entwickelt, so können sie eine syntaktische und semantische Prüfung aufrufen, die Fehler in dem entwickelten Struktogramm anzeigt, damit diese anschließend verbessert werden können. Da oftmals alternative Lösungsmöglichkeiten für eine gestellte Aufgabe existieren, müssen über die Tool-Box auch Fehler in unterschiedlichen Lösungswegen erkannt werden. Dazu wird zunächst eine statische Analyse durchgeführt. Dabei wird ein Syntaxbaum aufgebaut, der anschließend während der Ausführung von einem Interpreter genutzt wird, um weitere Fehler zu erkennen. Alle Fehler werden über gestufte Fehlermeldungen angezeigt und können von den Lernenden im vorher erstellten Struktogramm verbessert

werden. Dabei wird jeder Fehler möglichst genau im Struktogramm eingegrenzt, d.h. der Block, in dem der Fehler auftritt, wird farblich markiert.

Hat ein Struktogramm die Korrektheitsprüfung durchlaufen, wird es durch einen Interpreter an einem Beispiel animiert. Bei der Animation wird analog zu den Instruktionsanwendungen (vergleiche Abschnitt 3.2.1) das Struktogramm und parallel dazu das animierte Beispiel angezeigt. Beide Arten von Anwendungen unterscheiden sich nur dadurch, dass es sich bei dem einen um ein selbst entwickeltes Struktogramm handelt und daher Fehler auftreten können. Diese werden über das Feedback beschrieben, und es existiert eine Möglichkeit, um das Struktogramm erneut zu editieren. Ansonsten gelten die in Abschnitt 3.2.1 beschriebenen Anforderungen.

# 4 Umsetzung der Umgebung

## 4.1 Einstieg

Besuchern des Virtuellen Informatiklabors soll beim Eintritt in das Labor der Eindruck vermittelt werden, dass sie sich in einer explorativen Umgebung befinden, in der sie Themen je nach Interesse auswählen und sich damit beschäftigen können. Aus diesem Grund wurde das Labor basierend auf einer Raummetapher entwickelt, in der sich ein Besucher mithilfe der Maus bewegt und dabei den virtuellen Raum durchwandern kann (Kerres, 2001, S. 243ff; Haack, 2002, S. 130ff). In diesem Raum sind unterschiedliche Objekte plaziert, die jeweils einen Themenbereich repräsentieren. Bewegt sich die Maus über einzelne Objekte, so wird per Tooltip der Themenbereich angezeigt, über einen Mausklick wird dann direkt auf die Einstiegsseite des Themas verzweigt.


Abbildung 1: Raum des Virtuellen Informatiklabors

## 4.2 Orientierung und Navigation

Um eine intuitive Orientierung zu ermöglichen, wurde durchgängig für das gesamte Informatiklabor eine einfache Navigation mit folgenden Elementen entwickelt:

- **Home**: Link zu der Übersichtsseite mit allen Themen des Virtuellen Informatiklabors.
- **Aktuelles Thema**: Eine graphische Landkarte mit Fischaugensicht gibt einen Überblick über das aktuelle Themengebiet. Über diese Landkarte werden die zentralen Informationselemente in der näheren Umgebung direkt verlinkt.
- **Lernwege**: Abhängig vom Schwierigkeitsgrad und dem Vorwissen der Lernenden soll es möglich sein, das Themengebiet über einzelne Lernwege mit einer Vorwärts- und Rückwärtsnavigation zu erschließen. Dabei sind die Lernwege als Vorschläge zu verstehen, in welcher Reihenfolge die Anwendungen aufeinander aufbauend bearbeitet werden können. Lernwege sind ein Zusatzangebot neben der selbstbestimmten Auswahl von Anwendungen über die graphische Landkarte. Es gibt momentan die beiden Lernwege "alle Inhalte" und "Experimente", wobei der letztere Weg nur

Anwendungen mit Konstruktionscharakter, also Experimente und Algorithmen zum Entwickeln, umfasst.

## 4.3 Benutzerunterstützung

Im Virtuellen Informatiklabor wird MIA, die MI-Assistentin (siehe Abbildung **2**), als Guide (persönliche Tutorin) eingesetzt um die Orientierung zu erleichtern, Themengebiete vorzustellen, die Lernenden zu motivieren, sie bei der Problemlösung zu unterstützen und um einen sozialen Kontext beim Lernen aufzubauen (Gulz, 2004).


Abbildung 2: Guide MIA – informativ, mit motivierendem Feedback, mit korrigierendem Feedback

MIA soll die Lernenden im Lernprozess unterstützen und dabei als inhaltlich kompetent, aber auch sympathisch empfunden werden. Beim Entwurf der Figur wurden Aspekte des ARCS-Modells berücksichtigt (ARCS-Modell: Akronym der vier motivationalen Aspekte Attention, Relevance, Confidence und Satisfaction), um die Motivation der Lernenden zu erhöhen (vergleiche die Empfehlungen von Niegemann et al. (2004, S. 208 ff.) zur Konkretisierung des ARCS-Modells (Keller, 1983, S. 386 ff.) in multimedialen Lernumgebungen).

Hintergrundinformation zu den behandelten Themen, zu Algorithmen und den Werkzeugen wurden prägnant beschrieben und in einem **Lexikon** abgelegt. Für das Lexikon wurde eine Darstellung als Buch gewählt, so dass eine Analogie zum Nachschlagen nach Information in einem realen Buch entsteht (vergleiche Abbildung **3**).

Auf jeder Seite des Virtuellen Informatiklabors existiert ein Button, mit dem direkt in das Lexikon gesprungen werden kann. Der Lernende muss dann aber über das Inhaltsverzeichnis selbständig das für ihn interessante Thema suchen. Weiterhin verweist MIA kontextsensitiv auf Inhalte im Lexikon, die zur Lösung der Aufgabe beitragen können. Hier entfällt die Suche für den Lernenden; es wird direkt auf die entsprechenden Seiten gesprungen.

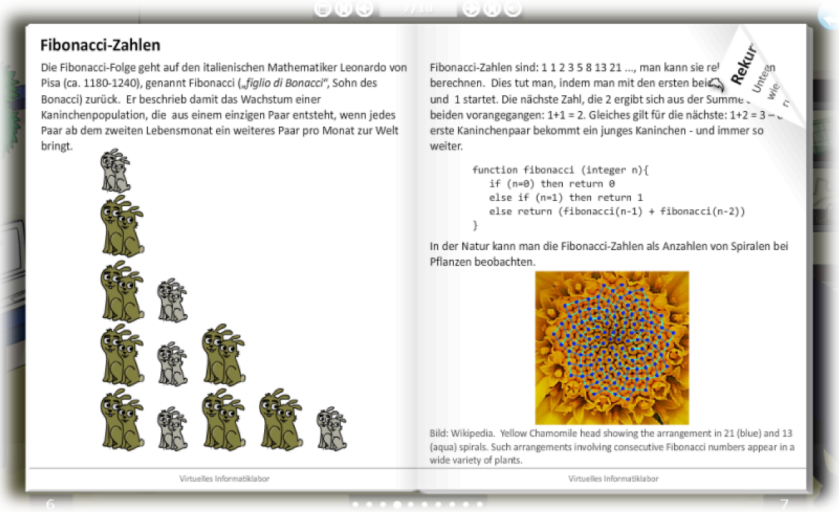


Abbildung 3: Lexikon des Virtuellen Informatiklabors

Zusätzlich besitzt jede Anwendung des Virtuellen Informatiklabors über einen Hilfebutton (mit dem "?"-Symbol gekennzeichnet) eine passive Hilfe, die jederzeit im Lernprozess zu Rate gezogen werden kann. Es handelt sich um eine kontextsensitive Hilfe, die auf die entsprechenden Seiten des Lexikons verlinkt ist. Wählt der Lernende die Hilfe an, so stehen ihm über konkrete Fragen mehrere inhaltliche Bereiche zur Auswahl. Konkret bietet die Hilfe (siehe Abbildung **4**):

- Informationen zum Thema – Beispiel: "Möchtest du mehr über Rekursion erfahren?",
- eine Erklärung des konkreten Algorithmus – Beispiel: "Oder dich über das Spiel Türme von Hanoi informieren? ",
- eine Beschreibung der Werkzeuge – Beispiel: "Oder willst Du wissen, wie Struktogramme aufgebaut sind? "
- und schließlich Tipps zum Lösen der Aufgabe – Beispiel: "Möchtest du einen Hinweis zur Lösung?"

Die Tipps zum Lösen der Aufgabe sind ebenso wie das Feedback auch gestuft aufgebaut und geben Schritt für Schritt mehr Hinweise auf die Lösung. So können auch Lernende mit unterschiedlichem Vorwissen schrittweise an die Lösung herangeführt werden.

# 5 Prototypische Umsetzung des Themas Rekursion

Als erstes Thema zur Umsetzung wurde die **Rekursion** gewählt, da dies zum einen ein wichtiges Prinzip beim Entwickeln von Algorithmen ist – man wendet das gleiche Prinzip wieder an – und zum anderen ist die Rekursion ein Thema, das gerade Anfängern Probleme bereitet (Boles, 2008). Aus diesen Gründen kann eine E-Learning-Anwendung, die unterstützend zum Verständnis und zum selbständigen Lernen eingesetzt wird, die Hemmschwelle bei Anfängern reduzieren, die Motivation erhöhen und einen höheren Lernerfolg bringen.

## 5.1 Aufbau des Themas Rekursion

Auf der Startseite zur Rekursion wird inhaltlich in das Thema eingeführt, indem wichtige Begriffe und die Bedeutung des Themengebietes für den Lernenden beschrieben werden. Aus didaktischer Sicht ist von besonderer Bedeutung, die Lernenden zu motivieren und damit für das Thema zu interessieren. MIA, der persönliche Tutor, führt in das Thema ein. Sie spricht den Lernenden direkt an und erzeugt dadurch eine Vertrautheit (siehe Abbildung **4**). Neben dem Einstiegstext werden kurze Beispiele präsentiert, um die abstrakten Begriffe und Inhalte in einen vertrauten Kontext zu stellen.

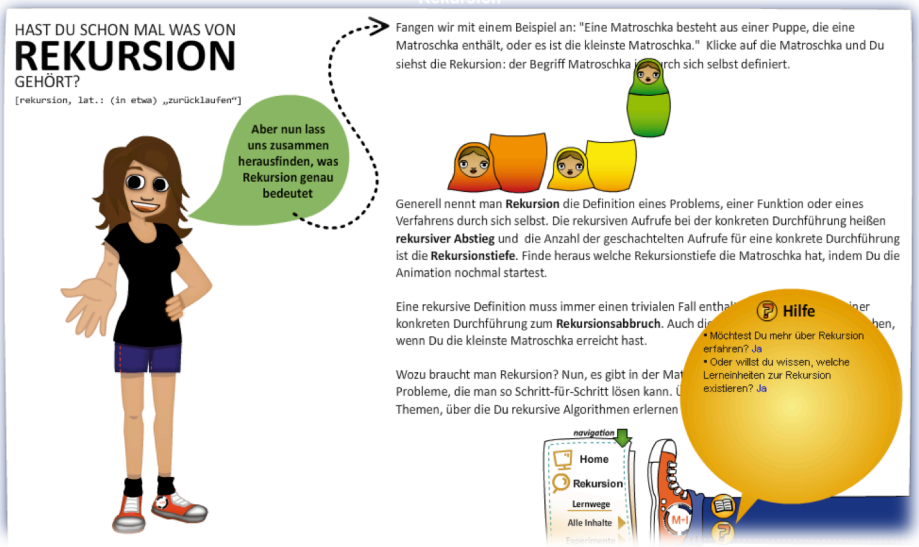


Abbildung 4: Einstiegsseite zur Rekursion und Hilfebutton

Die Strukturierung der Lerneinheit Rekursion basiert auf der Auswahl von geeigneten Algorithmen, die auf einer rekursiven Lösung basieren. Dabei sollen mehrere Algorithmen integriert werden, wobei der Schwierigkeitsgrad steigt. Dieser Ansatz vermeidet, dass die Lernenden nur einen Lösungsweg für einfache (mathematische) Rekursionen erlernen und vermuten, dass dieser Weg auf alle Rekursionen angewendet werden kann (Scholtz & Sanders, 2010). Zusätzlich werden unterschiedliche Sichtweisen auf den Ablauf der Rekursion (rekursiver Abstieg, rekursiver Aufstieg) geboten, damit die Lernenden ein korrektes mentales Modell der Rekursion entwickeln können. Aus der Forschung ist bekannt, dass besonders der rekursive Aufstieg für viele Lernende schwer zu verstehen ist (Sanders, Galpin & Götschi, 2006).

Weit verbreitet sind lineare und kaskadenartige Rekursion, die auch im Virtuellen Informatiklabor erlernt werden sollen. Dabei ist die lineare Rekursion einfacher zu verstehen, da die lineare Aufrufstruktur deutlich einfacher ist als die Baumstruktur einer kaskadenartigen Rekursion. Sie dient als Einstieg und soll zuerst grundlegend erlernt werden. Mit diesem Basiswissen ist dann ein Verstehen der kaskadenartigen Rekursion besser möglich.

Daher wurde zunächst als einfaches Beispiel für eine lineare Rekursion die **Fakultätsfunktion** gewählt. Die Fakultätsfunktion ist zwar ein klassisches Beispiel bei der Vermittlung der Rekursion und deswegen den meisten Schülern aus der Mathematik bekannt, aber da es sich um eine mathematische Funktion handelt, nicht unbedingt motivierend. Diese Motivation soll nun über das konkrete Anwendungsbeispiel "Lotto spielen" als Einstieg gefunden werden, denn man kann auch heute noch davon ausgehen, dass viele Schülerinnen und Schüler das Lottospiel aus dem Fernsehen oder dem Internet kennen und sich deshalb dafür interessieren, wie groß die Gewinnchancen dabei sind. Für die Fakultätsfunktion spricht außerdem ihr einfacher rekursiver Algorithmus, der ein erstes mentales Modell einer linearen Rekursion bei den Lernenden unterstützt.

Als zweites Thema wurde das bekannte Knobelspiel **"Türme von Hanoi"** ausgewählt, dessen Rekursion einen höheren Schwierigkeitsgrad aufweist als die Fakultätsfunktion. Gleichzeitig weckt dieses Spiel aber bei den Lernenden das Interesse, eine schnelle und effiziente Lösung zu finden, und selbst jüngere Lernende sind in der Lage, einen eigenständigen Lösungsweg zu entwickeln (Gunion, Milford & Stege, 2009), da die rekursive Lösungsidee für die Türme von Hanoi sehr einfach zu formulieren ist.

Bei der **Fibonacci-Folge,** dem dritten Algorithmus, handelt es sich um eine kaskadenartige Rekursion, die mit vielen Beispielen aus der Natur visualisiert werden kann. Der rekursive Ablauf ist durch die Summation von zwei rekursiven Aufrufen schwieriger zu verstehen als die bisherigen Beispiele. Daher wird über dieses Beispiel ein Schwerpunkt auf die Vermittlung der baumartigen Aufrufstruktur gelegt.

Für die drei erwähnten Algorithmen werden verschiedene Anwendungen angeboten, die in die folgenden Kategorien eingeordnet werden können (Tabelle **2**):

- Instruktion (Erlernen des Algorithmus),
- Experiment (Algorithmus anwenden)
- Struktogramm entwickeln (Algorithmus selbst entwickeln).

## 5.2 Instruktion: Fakultät & Matroschkas

Zu jedem Algorithmus existiert im Virtuellen Informatiklabor eine Instruktionsanwendung, mit deren Hilfe der Algorithmus schrittweise erlernt werden kann. Abbildung **5** zeigt die Instruktionsanwendung zur Berechnung der Fakultätsfunktion. Als Interaktionen stehen den Lernenden die Auswahl des Eingabeparameters, in diesem Fall die Wahl des Parameters n, für den die Fakultät berechnet werden soll, sowie die Steuerung des Algorithmus über die Kontrollleiste zur Verfügung. Speziell der Einzelschrittmodus in Vorwärts- und Rückwärtsrichtung unterstützt das Lernen des jeweiligen Algorithmus, da jederzeit zu dem Punkt gewechselt werden kann, der eine nochmalige Erklärung erfordert.

Auf der rechten Seite ist ein Struktogramm mit dem rekursiven Algorithmus der Fakultätsfunktion dargestellt. Während der Algorithmus im Play- oder Einzelschrittmodus abläuft, wird in den Struktogrammen der aktuelle Ausführungsschritt markiert. Die Rekursion wird über die in der Aufrufreihenfolge übereinanderliegenden Struktogramme mit den entsprechenden Aufrufparametern visualisiert.

Um diese formale Darstellung den Lernenden nahe zu bringen, wird parallel dazu auf der linken Seite in einer Animation die Analogie der Matroschkas verwendet, weil sie zum einen ein anwendungsorientiertes Einleitungsbeispiel ist und zum anderen eine hilfreiche Visualisierung einer linearen Rekursion darstellt. Jede Matroschka steht für den Aufruf der Fakultätsfunktion mit einem bestimmten Parameter n. Dies wird über die Beschriftung fac(n) auf dem Label, den jede Matroschka in den Händen hält, deutlich gemacht. Diese Label können gleichzeitig über die Hintergrundfarbe den farblich entsprechenden Struktogrammen zugeordnet werden. Rekursionsabstieg und -aufstieg sind so parallel über das Bild der ineinander verschachtelten Matroschkas und den übereinanderliegenden rekursiven Aufrufen der Struktogramme visualisiert. Unterhalb der Matroschkas wird textuell der Ablauf der Rekursion über die entsprechenden Formeln angegeben.

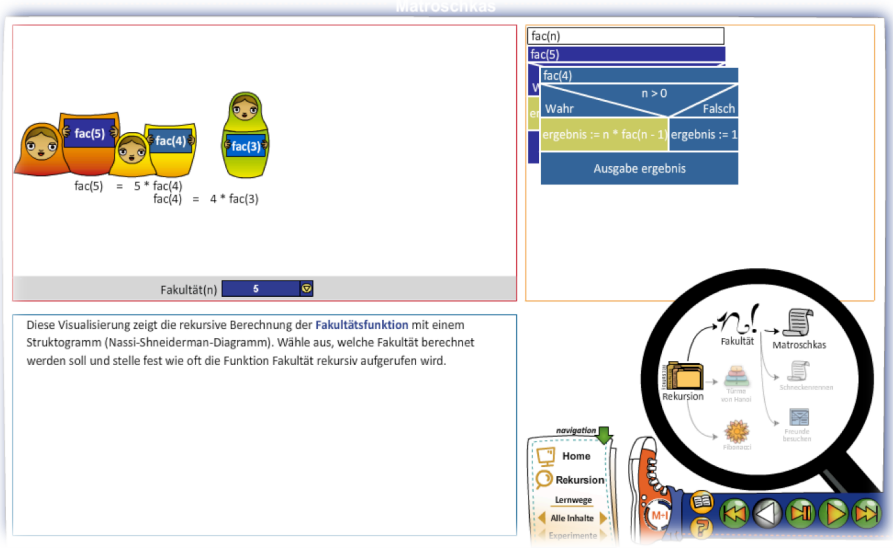


Abbildung 5: Instruktion zur Berechnung der Fakultätsfunktion und Fischaugensicht

Damit der rekursive Algorithmus, der im Struktogramm dargestellt ist, Schritt für Schritt verstanden werden kann, verfügt die Anwendung im Einzelschrittmodus über Lehrtexte, die als Tooltip jeden Block des Struktogramms kurz erklären. Über dieses Konzept kann zusätzliche Information abgerufen werden, um die einzelnen Elemente des Struktogramms besser zu verstehen.

## 5.3 Experiment: Türme von Hanoi

Für das Knobelspiel "Die Türme von Hanoi" wurde das in Abbildung **6** dargestellte Experiment entwickelt. Die Lernenden haben zu Beginn die Möglichkeit die Anzahl der Ringe (zwischen 3 und 6) auszuwählen und die Start- und Zielposition über Anklicken der Bereiche zu bestimmen. Über diese Wahl der Startparameter kann zum einen der Schwierigkeitsgrad variiert werden und zum anderen wird eine gewisse Abwechslung erreicht, da die Türme zwischen unterschiedlichen Positionen zu bewegen sind.

Das Experiment kann nun von den Lernenden einfach ausprobiert werden, indem schrittweise ein Ring per Drag-and-Drop von einer Position zu einer anderen bewegt wird, oder aber es wird gezielt nach einer Lösungsstrategie vorgegangen. In der Arbeitsanweisung wird empfohlen eine rekursive Lösungsidee zu versuchen, die in dem oberen rechten Bereich (blauer Rahmen) textuell skizziert ist. Diese Lösungsidee lässt klar die rekursiven Aufrufe und auch die Bedingung für den Rekursionsabbruch erkennen.

Lernende müssen sich jedoch nicht an die vorgegebene rekursive Lösungsidee halten, und da es sich um ein freies Experiment handelt, wird jeder mögliche Lösungsweg zugelassen. Bei jedem Schritt wird nur über einen Zähler angegeben, wie viele Schritte bisher benötigt wurden. Wird eine Lösung erzielt, so erscheint eine Feedbackmeldung über MIA, die bewertet, ob die Lösung mit der minimalen Anzahl von Schritten erreicht wurde. Dieses verzögerte Feedback erlaubt es, eigene Lösungswege zu durchlaufen und nicht über frühere Rückmeldungen auf einen vorgegebenen Weg gezwungen zu werden.

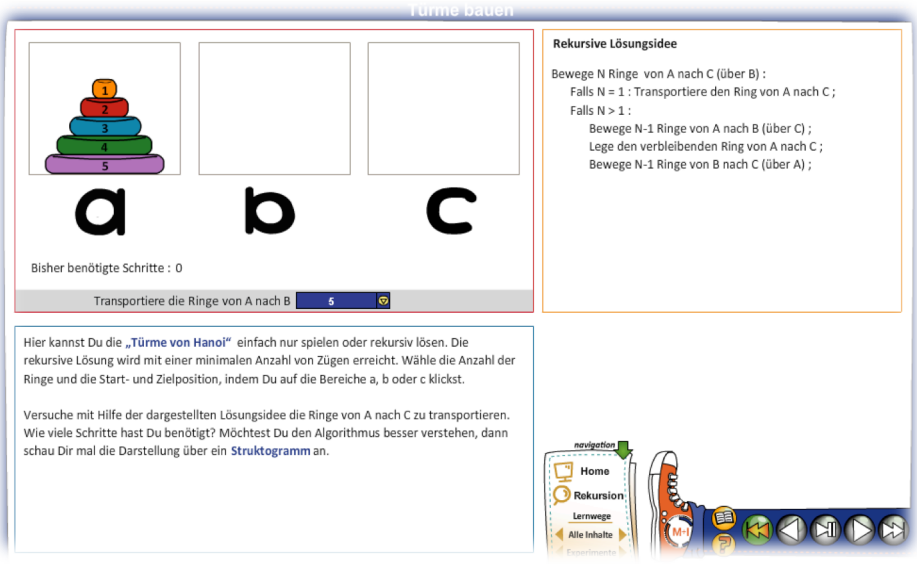


Abbildung 6: Experiment Türme von Hanoi

## 5.4 Struktogramm entwickeln: Freunde besuchen

Als Anwendung zum Fakultät-Algorithmus wurde das Anwendungsbeispiel "Freunde besuchen" integriert, in dem Lernende mit Hilfe der Tool-Box selbst ein einfaches Struktogramm entwickeln sollen. In dem Anwendungsbeispiel sollen die möglichen Fahrtrouten berechnet werden, um m von n Freunden zu besuchen. Diese Aufgabenstellung und eine Kurzfassung zur Bedienung der Tool-Box wird im Bereich der Aufgabenstellung (unten links) beschrieben (vergleiche Abbildung **7**). Dabei sind klar die Eingangs- und Ausgangsvariablen angegeben. Das Struktogramm zeigt zu Beginn nur den Kopf mit dem Aufruf und den Eingangsvariablen und einen Block mit der Ausgabe und der entsprechenden Variablen. Mit diesem Einstieg ist direkt vorgegeben, welche Bedeutung die einzelnen Variablen besitzen. In Abbildung **7** wurden bereits erste Blöcke zum Struktogramm hinzugefügt. Dazu wird das entsprechende Symbol aus der Spalte Strukturblöcke ausgewählt und auf die Position im Diagramm gezogen, und anschließend kann der Block über die Editorfunktion mit Inhalten gefüllt werden. Betrachtet man den Bereich "Editiere den Strukturblock" in Abbildung **7**, so sieht man, dass sowohl die Variablen, als auch die Funktion besuche hier über Buttons

ausgewählt werden können. Die Tool-Box kann generisch verwendet werden, über XML (Extensible Markup Language) können die benötigten Eingabeparameter und Funktionen konfiguriert werden, und darauf aufbauend werden im GUI die entsprechenden Buttons eingefügt. Somit sind weitere Applikationen über die Vorkonfiguration einfach zu erstellen.

Generell wurden die genutzten Strukturblöcke und die benötigten Variablen und Funktionen so einfach wie möglich gehalten, um die Lernenden nicht zu überlasten. Gleichzeitig sind die Lernenden aber in ihren Interaktionen nicht eingeschränkt – sie können mit den vorhandenen Hilfsmitteln experimentieren und ihre eigene Lösung entwickeln.

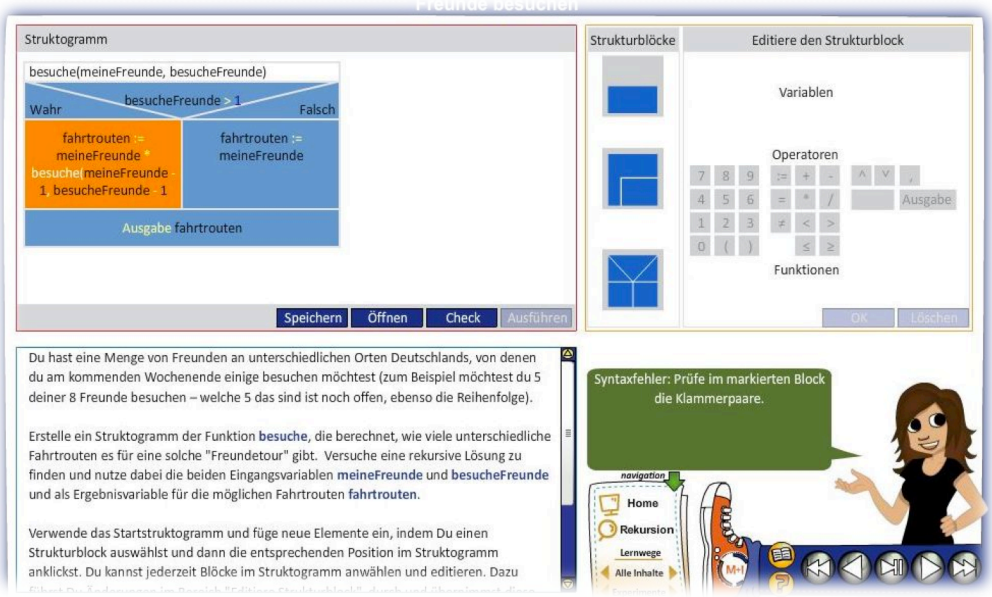


Abbildung 7: Besuche Freunde – ein Struktogramm entwickeln mit einer Fehlermeldung für einen Syntaxfehler

Ist ein Struktogramm erstellt, so kann dieses mit dem Check-Button auf Korrektheit geprüft werden. Diese Prüfung sichert zunächst über einen Syntaxcheck die syntaktische Korrektheit und findet anschließend über sogenannte Assertions – das sind Zusicherungen im Programmcode, die immer gelten müssen – semantische Fehler. Für jeden Fehler werden über MIA die gestuften Fehlermeldungen ausgegeben und gleichzeitig im Struktogramm der betroffene Block markiert (siehe Abbildung **8**). Der Lernende kann den markierten Block selektieren und anschließend versuchen, den Fehler zu verbessern.

Wenn ein syntaktisch korrektes Struktogramm vorliegt, kann dieses visuell animiert werden, während parallel die Programmschritte im Struktogramm visualisiert werden.

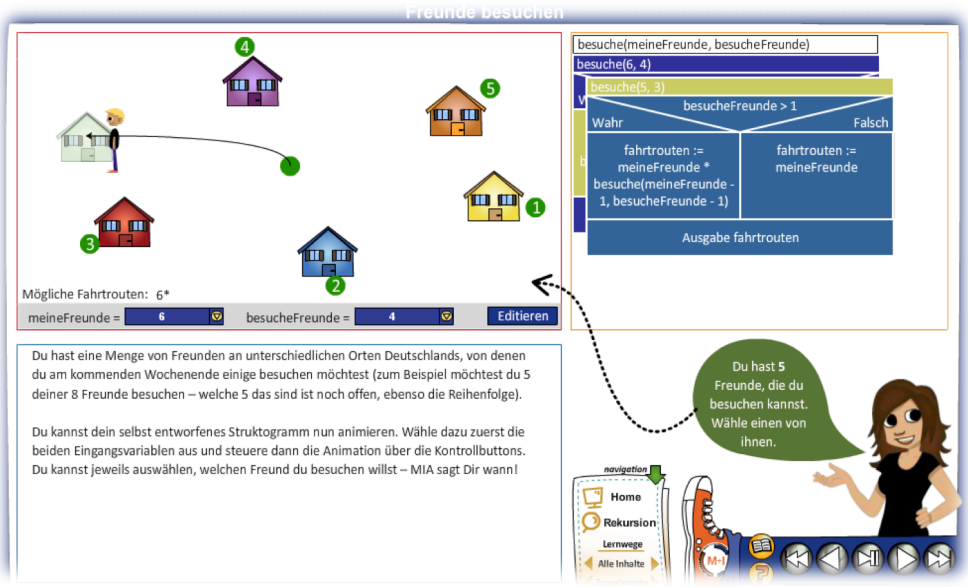


Abbildung 8: Besuche Freunde - Animation eines entwickelten Struktogramms

# 6 Evaluation des Virtuellen Informatiklabors

Im Wintersemester 2011/2012 wurde im Rahmen einer Informatik-Einführungsvorlesung eine Studie zur Usability und dem Lernerfolg mit dem Prototyp des Virtuellen Informatiklabors durchgeführt.

Der Usability-Test wurde mit 6 Versuchspersonen durchgeführt, die zunächst einige vordefinierte Aufgaben am Rechner durchführen sollten und nachfolgend zum Umgang mit dem System befragt wurden. Wie erwartet ergab dieser Test durchaus Verbesserungsbedarf, obwohl das Design als sehr ansprechend bewertet wurde. Der nachfolgende Lernerfolgstest war erfolgreich, wobei die Ergebnisse mit einer verbesserten Usability sicherlich noch positiver werden.

Der Lernerfolgstest wurde mit 83 Studierenden durchgeführt, 34 männliche und 49 weibliche. Diese wurden zunächst im Plenum kurz in das Thema Rekursion eingeführt. Anschließend bearbeiteten immer zwei Studierende an einem Rechner vordefinierte Aufgaben des Virtuellen Informatiklabors. Direkt im Anschluss war ein Fragebogen auszufüllen mit einem Teil Wissens- und Verständnisfragen und einem zweiten Teil, in dem die Studierenden selbst einschätzen sollten, was und wie viel sie erlernt hatten. Insgesamt standen 90 Minuten für den Test zur Verfügung.

Vor dem Test wussten nach eigenem Bekunden nur 12,5 Prozent der Beteiligten, was Rekursion ist. Danach hatten 63 Prozent der weiblichen und fast 83 Prozent der männlichen Teilnehmer verstanden, worum es bei der Rekursion geht. Den rekursiven Algorithmus zur Berechnung der Fakultät einer Zahl hatten beispielsweise fast 62 Prozent verstanden. 83 Prozent hatten anhand eines Beispiels verstanden, was rekursiver Ab- und Aufstieg bedeutet. 85 Prozent der Befragten würden die Lektion weiterempfehlen.

Etwa fünf Wochen später wurden sechs Freiwillige noch einmal ausführlich befragt. Die unterschiedliche Aufbereitung der Inhalte wurde dabei ebenso begrüßt, wie der explorative Lernansatz für den Einstieg in ein Thema oder ein Kapitel. Die Animationen visualisieren mit ihrer Einzelschrittfunktion den rekursiven Auf- und Abstieg gut und wurden daher als motivierend empfunden. Von den Anwendungsarten *Experiment* und *Algorithmen entwerfen* wünschen sich die Befragten noch weitere Beispiele, am besten mit aufsteigendem Schwierigkeitsgrad.

# 7 Zusammenfassung & Ausblick

Das vorgestellte Virtuelle Informatiklabor wurde konzipiert, um Schüler und Studierende Algorithmen der Informatik in einer konstruktivistisch geprägten Lernumgebung aktiv und selbständig erlernen zu lassen. Es geht weit über die verbreiteten Algorithmen-Visualisierungen hinaus, da es neben den Instruktionsanwendungen zum Erlernen der Algorithmen zusätzlich Experimente und konstruktivistische Anwendungen enthält, die ein aktives und selbständiges Lernen ermöglichen. Alle Anwendungen zeichnen sich durch einen hohen Grad an Interaktivität aus und streben damit eine höhere Lern- und Behaltensleistung an. Die Elemente der Lernumgebung wurden didaktisch und methodisch mit dem Ziel der Verbesserung des Lernerfolges entwickelt. Beispielsweise werden konsequent ein kontextsensitives Feedback und eine kontextsensitive Hilfe eingesetzt, um Lernende im Lernprozess zu unterstützen, und die Motivation und der Transfer des Gelernten auf andere Bereiche werden über anwendungsorientierte Beispiele und multiple Perspektiven auf einen Algorithmus angesprochen. Design und Sprache wurden an die Zielgruppe angepasst und sollen damit eine spontane Akzeptanz der Lernumgebung fördern.

Parallel zum Entwurf des Virtuellen Informatiklabors wurde Rekursion als erstes Thema konzipiert und dabei mehrere Anwendungen zu diesem Themenbereich entwickelt und umgesetzt. Seit dem Sommer 2012 existiert auch das Thema Listen. Im Rahmen dieser Realisierungen wurde eine Tool-Box zur selbständigen Erstellung von Struktogrammen entworfen, implementiert und an ersten Beispielen erprobt. Dieses flexible Werkzeug ermöglicht ein selbstgesteuertes, entdeckendes Erlernen von Algorithmen. Erste Untersuchungen zum Lernprozess und Lernerfolg wurden durchgeführt, weitere sind geplant und bilden die Basis für Weiterentwicklungen. Das Virtuelle Informatiklabor ist unter der Adresse http://mi-learning.hs-offenburg.de/mi-vil/ zu finden.

Die nächsten Themenbereiche, die realisiert werden, sind das Programmieren lernen und weitere grundlegende Algorithmen wie Such- oder Sortieralgorithmen. Speziell der Bereich Programmieren lernen kann von den bisherigen Erfahrungen profitieren, da hier für die konstruktivistischen Anwendungen eine Tool-Box mit einfachen Elementen geplant ist, die ähnlich zu der Tool-Box für Struktogramme funktioniert.

Das Virtuelle Informatiklabor soll im Endausbau mit einer Vielzahl von Algorithmen und Methoden der Informatik für Lernende eine motivierende Lernumgebung bilden, die flexibel über das Internet genutzt werden kann. Damit können selbstgesteuert die entsprechenden Lerninhalte wiederholt und experimentell geübt werden.

**Danksagung**

Ein herzliches Dankeschön an Carolina Bernal für die Unterstützung bei der Realisierung der Videos und der HTML-Version des Artikels.

# 8 Literatur

Anderson, L.W., Krathwohl, D.R. (Eds.). (2001). A Taxonomy for Learning Teaching and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives. Addison Wesley Logman.

Arnold, R., Hartmann, W., Reichert, R. (2005). Entdeckendes Lernen im Informatik-Unterricht. Proceedings of INFOS 2005 "Unterrichtskonzepte für informatische Bildung". 11. Fachtagung Informatik und Schule der Gesellschaft für Informatik (GI), Dresden, 28.-30. September 2005.

Ben-Ari, M. (1998). Constructivism in Computer Science Education. In D. Joyce & J. Impagliazzo (Eds.), Proceedings of the Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education - SIGCSE '98 (pp. 257-261). New York. NY. USA: ACM.

Ben-Ari, M. (2002). From Theory to Experiment to Practice. In CS Education. 2nd Annual Finnish/Baltic Sea Conference on Computer Science Education. Koli, Finland. 19 October, 2002.

Boles, D. (2008). Programmieren spielend gelernt mit dem Java-Hamster-Modell. 4. Auflage. Wiesbaden: Vieweg +Teubner.

Brinda, T., Fothe, M., Friedrich, S., Koerber, B., Puhlmann, H., Röhner, G., Schulte, C. (2008). Grundsätze und Standards für die Informatik in der Schule, Bildungsstandards Informatik für die Sekundarstufe I, Gesellschaft für Informatik. Berlin: LOG IN Verlag. Zugriff am 26.9.2012 unter http://www.gi.de/fileadmin/redaktion/empfehlungen/Bildungsstandards_2008.pdf.

Faltin, N. (1999). Gestaltung von Lernprogrammen zu Algorithmen für aktives Lernen mit virtuellen Brettspielen. In Gesellschaft für Informatik, Tagungsband "Informatiktage 1999" (pp. 92-95). 12. - 13. November 1999 in Bad Schussenried: Konradin Verlag

Fuller, U., Johnson C. G., Ahoniemi, T., Cukierman, D., Hernán-Losada, I., Jackova, J., Lahtinen, E., Lewis, T. L., McGee Thompson, D., Riedesel, C., Thompson, E. (2007). Developing a Computer Science-Specific Learning Taxonomy. SIGCSE Bulletin. Volume 39 Issue 4, 152-170.

Gallenbacher, J. (2008). Abenteuer Informatik: IT zum Anfassen – vom Routenplaner bis Online Banking. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag.

Gulz, A. (2004). Benefits of Virtual Characters in Computer Based Learning Environments: Claims and Evidence. International Journal of Artificial Intelligence in Education, Volume 14 Issue 3,4, 313-334.

Gunion, K., Milford, T., Stege, U. (2009). Curing Recursion Aversion. Proceedings of the 14th annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education. SIGCSE Bulletin Volume 41 Issue 3, 124-128.

Haack, J. (2002). Interaktivität als Kennzeichen von Multimedia und Hypermedia. In Issing L. J. & Klimsa P. (Eds.), Information und Lernen mit Multimedia (3. Auflage, S. 127-136). Weinheim: Beltz PVU.

Hanselmann, U. (2009). Die große Kraft. In Die Zeit vom 20.5.2009, Nr. 22, S. 74.

Helminen, J., Malmi, L. (2010). Jype - a Program Visualization and Programming Exercise Tool for Python. In Proceedings of the 5th International Symposium on Software Visualization - SOFTVIS '10 (pp. 153-162). New York, NY, USA: ACM.

Heublein, U., Hutzsch, C., Schreiber, J., Sommer, D., Besuch, G. (2009). Ursachen des Studienabbruchs in Bachelor- und in herkömmlichen Studiengängen. Ergebnisse einer bundesweiten Befragung von Exmatrikulierten des Studienjahres 2007/08. HIS Hochschul-Informations-System GmbH (Hrsg.). Zugriff am 24.1.2011 unter http://www.his.de/pdf/21/studienabbruch_ursachen.pdf.

Hundhausen, C. D. (1998). Toward Effective Algorithm Visualization Artifacts: Designing for Participation and Negotiation in an Undergraduate Algorithms Course. In CHI 98 Conference Summary on Human Factors in Computing Systems - CHI '98 (pp. 54-55). New York, NY, USA: ACM.

Hundhausen, C. D., Douglas, S. A., Stasko, J. T. (2002). A Meta-Study of Algorithm Visualization Effectiveness. Journal of Visual Languages and Computing, Volume 13, Issue 3, 259-290.

Keller, J. (1983). Motivational Design of Instruction. In C. Reigeluth, (Ed.). Instructional Design Theories and Models. Hillsdale, NJ: Erlbaum.

Kerres, M. (2001). Multimediale und telemediale Lernumgebungen. Konzeption und Entwicklung. 2. Auflage. München: R. Oldenbourg Verlag.

Naps, T. L. (2005). JHAVÉ: Supporting Algorithm Visualization. IEEE Computer Graphics and Applications Volume 25 Issue 5, 49-55.

Niegemann, H., Hessel, S., Hochscheid-Mauel, D., Aslanski, K., Deimann, M., Kreuzberger, G. (2004). Kompendium E-Learning. Berlin: Springer-Verlag.

Pfannstiel, J., Sänger, V. Schmidt, C. (2009). Game-based Learning im Bildungskontext einer Hochschule – ein Praxisbericht. MedienPädagogik Themenheft Nr. 15/16: Computerspiele und Videogames in formellen und informellen Bildungskontexten. Zugriff am 25.1.2011 unter http://www.medienpaed.com/15/pfannstiel0904.pdf.

Rushkoff, D. (2010). Program or be Programmed – Ten Commands for a Digital Age. OR Books, New York.

Sanders I., Galpin, V., Götschi, T. (2006). Mental Models of Recursion Revisited. In Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education - ITICSE '06 (pp. 138-142). SIGCSE Bulletin. Volume 38, Issue 3.

Scholtz, T. L., Sanders, I. (2010). Mental Models of Recursion: Investigating Students' Understanding of Recursion. In Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE '10 (pp. 103-107). New York, NY, USA: ACM.

Shaffer, C., Cooper, M. L., Alon, A. J. D., Akbar, M., Stewart, M., Ponce, S., Edwards, S. H. (2010). Algorithm Visualization: The State of the Field. ACM Transactions on Computing Education (TOCE) Volume 10 Issue 3, Article 9.

Sänger, V., Schmidt, C. (2007). MI-Learning: ein Rahmenwerk für webbasiertes E-Learning. In Studienkommission für Hochschuldidaktik an Fachhochschulen in Baden-Württemberg (Hrsg.): Die Energie der Didaktik - Beiträge zum 7. Tag der Lehre, (S. 64-67), Hochschule Biberach.

Sänger, V., Schmidt, C. (2010). Erfahrungen mit einem hybriden Lernarrangement in der Informatik. Hamburger eLMagazin, Ausgabe #4 eLearning in den Naturwissenschaften.

Schmidt, C., Sänger, V., Endres, J. (2009). Hybride Lernarrangements - Informatik-Lehre an der Hochschule Offenburg", In A. Schwill, N. Apostolopoulos (Hrsg.), Lecture Notes in Informatics, DeLFI 2009 - die 7. E-Learning Fachtagung Informatik (S. 139-150), Berlin. Bonn: Gesellschaft für Informatik.

Strzebkowski, R., Kleeberg, N. (2002). Interaktivität und Präsentation als Komponenten multimedialer Lernanwendungen. In Issing L. J., Klimsa P. (Eds.), Information und Lernen mit Multimedia (3. Auflage, S. 229-246). Weinheim: Beltz PVU.

Tergan, S.-O. (2002). Hypertext und Hypermedia: Konzeption, Lernmöglichkeiten, Lernprobleme und Perspektiven. In Issing, L. J., Klimsa, P. (Eds.), Information und Lernen mit Multimedia (3. Auflage, S. 99-112). Weinheim: Beltz PVU.

Wedekind, J. (2012). Programmieren für Alle?! konzeptblog. Abgerufen am 26.9.2012 unter http://konzeptblog.joachim-wedekind.de/.

Weidner, I. (2011). Junginformatiker wollen alle haben. In Computerwoche, Heft 47, S. 36-39, Verlag IDG Business Media GmbH, München