**Peter Junglas**

# Simulationsprogramme für die Lehre – Was kommt nach Applets?

**Auszug.** Dass Java-Applets von den Browsern nicht mehr unterstützt werden, hat viele Entwickler, die Simulationsprogramme für die Lehre erstellen, gezwungen, ihre Programme nach JavaScript zu portieren. Für das PhysBeans-Projekt stellt dies allerdings eine besondere Herausforderung dar. Daher werden hier alternative Möglichkeiten untersucht, die Ergebnisse des Projekts weiterhin verfügbar zu machen.

## Physikalische Simulationen mit Java-Applets

In der Physikausbildung ist der Einsatz von Simulationsprogrammen seit langem etabliert. Stellvertretend für die riesige Anzahl frei verfügbarer Programme sei hier auf umfangreiche Projekte wie Physlet [1] und das verwandte EJS ("Easy Java Simulations") [2], das PhysBeans-Projekt [3] des Autors sowie auf die erstaunlichen Sammlungen [4, 5] verwiesen.

Zur Implementierung wurden dabei aus guten Gründen Java-Applets verwendet: Java stellt alle benötigten Konstruktionen und eine standardisierte Bibliothek für numerische und graphische Funktionen zur Verfügung, die Programme sind unabhängig vom Rechnertyp oder Betriebssystem, und die Applets lassen sich bequem in Webseiten integrieren, die Handlungsanweisungen oder Hintergrundinformationen enthalten.

Im Zuge der Umstellung auf HTML 5 und JavaScript haben aber die Browser-Hersteller seit 2015 die Unterstützung von Plugins im allgemeinen – und damit insbesondere von Java-Applets – eingestellt. Mit der Version Java 9 wurden dann die appletbezogenen Klassen in der Sprachdefinition als veraltet bezeichnet und sollen in zukünftigen Versionen wegfallen. Ein weiteres Problem entstand durch die zunehmende Verbreitung mobiler Endgeräte mit Android- oder iOS-Betriebssystem, auf denen nur Java-Grundfunktionen unterstützt werden, so dass Applets dort nicht laufen.

Als Ersatz für Java-Applets mit der Möglichkeit, Simulationen wie bisher direkt in Webseiten einbinden zu können, kommt zunächst nur JavaScript in Frage. Die Portierung einer großen Codebasis von Java nach

JavaScript ist allerdings sehr aufwändig, da die Sprachkonstrukte und Bibliotheken, vor allem zur Implementierung einer graphischen Benutzerschnittstelle, in beiden Sprachen sehr unterschiedlich sind. Trotzdem haben die oben aufgeführten Projekte – mit Ausnahme von PhysBeans – inzwischen weitgehend auf JavaScript umgestellt: Das EJS-Projekt heißt nun "Easy Java/JavaScript Simulations" und erzeugt seit Version 5 Java- und JavaScript-Programme [6], die dritte Auflage von [1] (inzwischen online vertrieben [7]) verwendet EJS 5 und enthält JavaScript-Versionen der bisherigen Programme, und auch die Applets der beiden Sammlungen [4, 5] wurden manuell bzw. im Rahmen des SwingJS-Projekts [8] konvertiert.

Die Konvertierung von PhysBeans scheiterte dagegen bisher nicht nur an den fehlenden Ressourcen, sondern vor allem an der etwas anderen Zielsetzung des Projekts: Es stellt einen Baukasten zur Verfügung, der es ermöglicht, ohne Programmierkenntnisse Applets für die eigenen Bedürfnisse zu konstruieren oder die vorhandenen Beispiele anzupassen. Es verwendet dazu das JavaBeans-Framework, um Komponenten zu erstellen, die in einer Entwicklungsumgebung graphisch manipuliert werden können. Eine analoge Infrastruktur existiert in JavaScript aber nicht. Im Folgenden sollen daher alternative Möglichkeiten untersucht werden, mit deren Hilfe die Ideen und Ergebnisse – und soweit wie möglich auch der Code – des PhysBeans-Projekts für moderne Browser und Endgeräte verfügbar gemacht werden können.

## Das PhysBeans-Projekt

Das PhysBeans-Projekt stellt ein Toolkit zur Verfügung, das die Erstellung von virtuellen Experimenten für physikalische (oder auch mathematische) Simulationen in der Form von Java-Applets ermöglicht. Es wurde 1998 begonnen, um Simulationen für die CD von [9] zu entwickeln, und wuchs im Lauf von 15 Jahren zu einem vielseitigen Werkzeug heran, mit dem auch didaktische Fragestellungen (etwa in [10]) oder numerische Probleme der Simulation (z. B. in [11]) untersucht wurden.

Inzwischen werden auf der Homepage des Projekts [12] über 160 Applets frei zur Verfügung gestellt, vor allem aus den Bereichen Mechanik, lineare Schwingungen (s. Abb. 1), nichtlineare Schwingungen und chaotische Systeme, Wellen, Optik (s. Abb. 2), Elektrodynamik (s. Abb. 3), Thermodynamik und Quantenmechanik (s. Abb. 4).

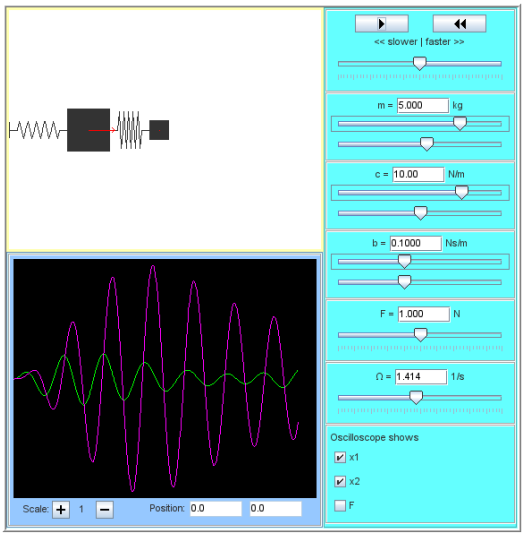


Abb. 1: Schwingerkette

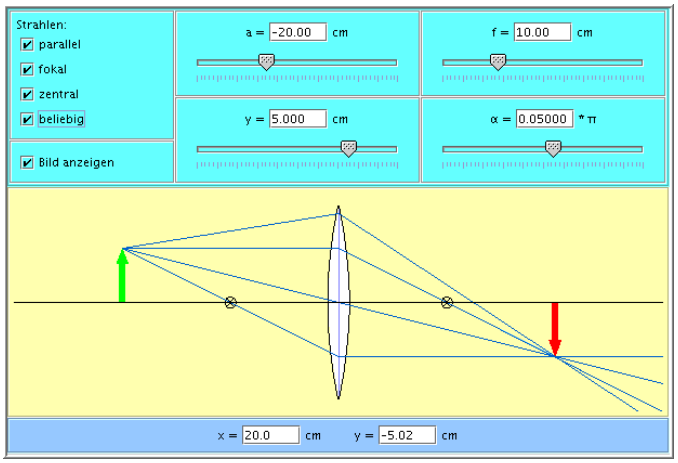


Abb. 2: Dünne Sammellinse

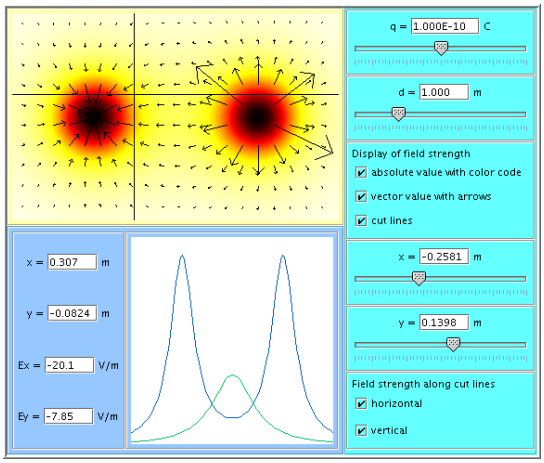


Abb. 3: Elektrischer Dipol

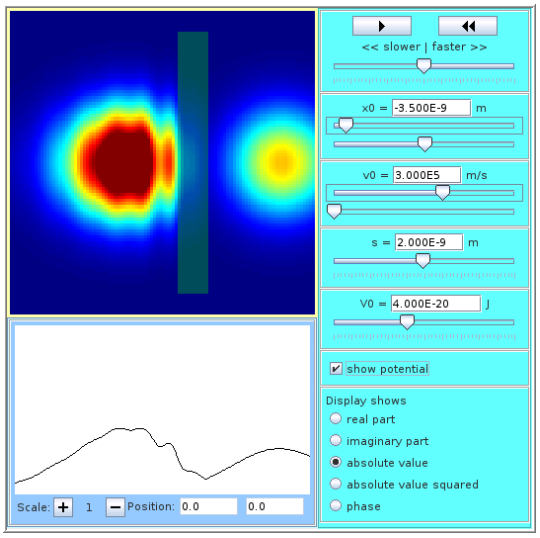


Abb. 4: Tunneleffekt

Die Basis des Toolkits ist die physbeans-Bibliothek, die eine große Zahl an Komponenten in der Form von JavaBeans [13] zur Verfügung stellt. Damit können in einer Entwicklungsumgebung auf graphische Weise eigene Simulationen erstellt werden, ohne dass vertiefte Java-Kenntnisse nötig sind. Ausgangspunkt der Entwicklung ist stattdessen ein Flussdiagramm, das alle Komponenten und die zwischen ihnen ausgetauschten Nachrichten darstellt (s. Abb. 5). Das genaue Vorgehen und viele Beispiele werden beschrieben in [3].

Der Java-Quellcode der Bibliothek umfasst etwa 85000 Zeilen. Dazu kommen knapp 50000 Zeilen für die damit erstellten Applets, die i. W. mit

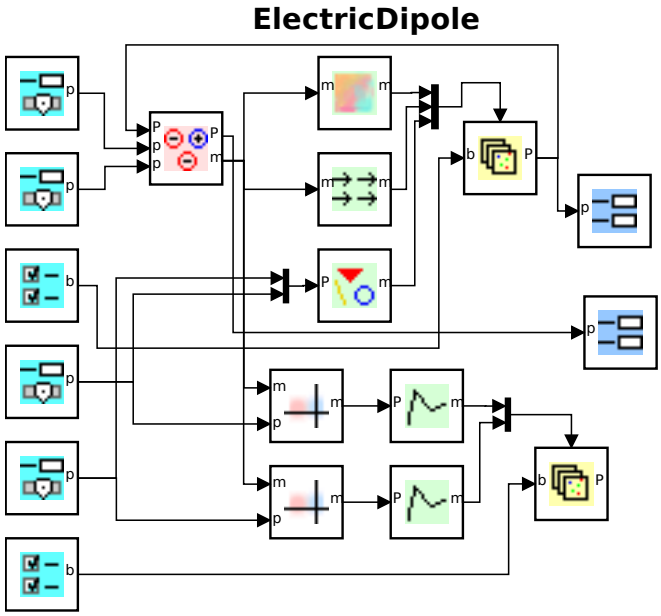


Abb. 5: Flussdiagramms des Applets “Elektrischer Dipol” (Abb. 3)

Hilfe der Entwicklungsumgebung aus dem graphischen Modell generiert wurden. Alle Quellen, das fertige jar-File sowie die Applets sind unter den üblichen Bedingungen auf der PhysBeans-Homepage [12] frei verfügbar.

## Maßnahmen mit geringen Quellcode-Änderungen

Ist man bereit, auf die direkte Einbindung des Applets in eine Webseite zu verzichten, gibt es eine einfache Möglichkeit, mit geringen Änderungen am Quellcode die PhysBeans-Programme weiter zur Verfügung zu stellen: die Java Web-Start-Technologie [14], eine Methode, um Java-Anwendungen über das Internet zu laden und zu starten. Aus der Sicht des Anwenders ändert sich nicht viel: Statt des eingebetteten Applets enthält die Webseite einen Link, nach dessen Anklicken die Java-Anwendung über das Netz geladen und in einem eigenen, vom Browser unabhängigen Fenster gestartet wird.

Damit dies funktioniert, müssen die Applets in Java-Applikationen umgeschrieben werden. Dazu bekommen sie eine `main`-Funktion, die die Aufgaben übernimmt, die sonst der Browser ausführt: Erzeugen eines Fensters (`JFrame`), Initialisieren (`init()`) und Starten des Applets (`start()`). Diese Funktionalität kann in eine globale Elternklasse ausgelagert werden, so dass die Änderungen an den einzelnen Applets marginal sind. Ein weiterer Unterschied besteht in der Art, wie Applets und Applikationen auf

Ressourcen, z. B. Bilddateien, zugreifen können, aber auch das lässt sich leicht anpassen. Auf diese Weise lassen sich Programme erzeugen, die sowohl als Applets als auch als Applikation gestartet werden können. Da die PhysBeans-Applets von Anfang so konzipiert worden sind, waren im konkreten Fall überhaupt keine Änderungen am Javacode nötig.

Da das Laden eines Programms aus dem Netz grundsätzlich ein hohes Risiko birgt, erfordert Java Web Start zusätzlich, dass man ein Zertifikat erstellt (oder kauft) und damit die Jar-Dateien signiert. Schließlich muss noch eine Startdatei (JNLP-Datei) für jedes zu startende Applet erstellt werden, auf das der Link der Webseite verweist. Dies alles ist wenig aufwändig und lässt sich leicht automatisieren.

Das klingt nach einer einfachen und bequemen Lösung: Abgesehen von der etwas schlechteren Verzahnung zwischen Text und Simulation laufen alle Programme bei minimalem Aufwand genau wie vorher. Aber das Verfahren hat auch Nachteile: Zunächst gibt es häufig Probleme mit der Sicherheitskontrolle, vor allem bei Verwendung eines selbst erstellten Zertifikats. Damit dies akzeptiert wird, muss man die Sicherheitsstufe auf dem Client-Rechner heruntersetzen, was je nach Betriebssystem einfach oder fast unmöglich ist. Vor allem aber benötigt der Anwender eine lokale Installation des Java-Laufzeitpakets. Dies lässt sich im Rahmen etwa einer Vorlesung vielleicht auf die Studenten abwälzen, ist aber dem allgemeinen Anwender in Zeiten komplett gekapselter Apps kaum noch vermittelbar. Aus diesem Grund hat Oracle den Java Web-Start-Mechanismus als "veraltet" eingestuft und verkündet, die Unterstützung in kommenden Versionen einzustellen [15].

## Maßnahmen mit großen Quellcode-Änderungen

Das Problem, für verschiedene Plattformen oder unterschiedliche Software-Umgebungen zu entwickeln, haben natürlich auch andere. Insbesondere bei der Spiele-Programmierung sind die Anforderungen sehr ähnlich, die Komplexität dagegen häufig wesentlich größer als bei physikalischen Simulationen. Daher wurden verschiedene Werkzeuge erstellt, die Spiele-Entwicklern eine plattform-unabhängige Programmierung mit komplexer Grafik und hoher Performance ermöglichen.

libGDX [16] ist ein solches Framework, das komplett auf Open-Source-Komponenten beruht. Es hat eine sehr aktive Nutzerschaft und

ist gut dokumentiert, u. a. durch mehrere Lehrbücher (z. B. [17]). Es ermöglicht eine Entwicklung komplett in Java, wobei graphische und andere "hardware-nahe" Funktionen durch Aufrufe der libGDX-Bibliothek ersetzt werden. Mit Hilfe diverser Tools werden die Programme dann automatisch auf andere Plattformen portiert, so dass gleichzeitig eine Java-Applikation, eine HTML5/JavaScript-Anwendung und Apps für Android und iOS erzeugt werden.

Für eine Portierung von PhysBeans müssen alle graphischen Komponenten für die libGDX-Plattform neu entwickelt werden, während sämtliche physikalischen, mathematischen und sonstigen Komponenten unverändert übernommen werden können. Da die Graphik-Funktionen in libGDX sehr systemnah sind, ist der Portierungsaufwand relativ hoch, dafür lässt sich das typische Aussehen aber auch recht detailliert nachbilden. Die schematische Darstellung physikalischer Objekte ("View-Beans") ist dabei noch vergleichsweise einfach, dagegen stellt die Nachbildung der Ein-/Ausgabe-Bausteine ("Input-/Output-Beans") eine größere Herausforderung dar. Dass dies grundsätzlich möglich ist, zeigt die erfolgreiche Portierung des Applets "Dünne Sammellinse" (Abb. 2), für das auf der PhysBeans-Homepage eine libGDX-basierte JavaScript-Version (Abb. 6) sowie eine Android-App zur Verfügung gestellt werden.

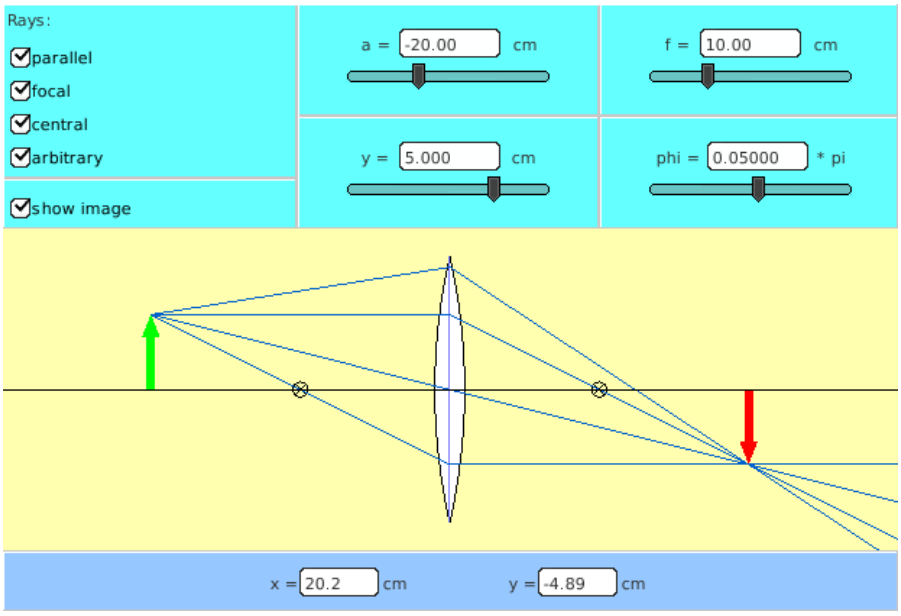


Abb. 6: JavaScript-Version des Applets "Dünne Sammellinse" (Abb. 2)

Eine vollständige Portierung von PhysBeans auf libGDX erfordert einige Anstrengung. Allerdings ist die Zahl verschiedener Input-/Output-Beans nicht groß, sie werden häufig wiederverwendet. Für die vielen View-

Beans lässt sich vermutlich durch geschicktes Einführen einer Zwischenschicht und geeignete objekt-orientierte Konstruktionen der Aufwand auf lange Sicht deutlich verringern. Dafür erhält man direkt vom Browser innerhalb einer Webseite ausführbare JavaScript-Versionen – und sogar Android- oder iOS-Apps, falls gewünscht – bei gleichzeitigem Erhalt des größten Teils der vorhandenen Codebasis. Vor allem aber bleibt für Nutzer von PhysBeans die Möglichkeit bestehen, physikalische Simulationsprogramme weitgehend ohne eigenen Java-Code zu erstellen.

## Komplette Neuentwicklung in JavaScript

Wenn es nur um die Portierung der im Rahmen des PhysBeans-Projekts erstellten Applets geht – ohne die Möglichkeit, weitere Applets mit graphischen Methoden zu erstellen –, kommt eine Komplett-Portierung nach JavaScript in Frage. Die Möglichkeiten von HTML5, insbesondere das Canvas-Element und die Web-Worker-API [18], sowie hilfreiche Bibliotheken wie Numeric Javascript [19] stellen zumindest die grundsätzlich benötigten Werkzeuge dafür zur Verfügung.

Außerdem ist die Performance von JavaScript-Anwendungen durch Verfahren wie Just-in-time-Kompilierung stark angestiegen. Ob das allerdings reicht für die hohen Anforderungen der PhysBeans-Applets zur Quantenmechanik (s. Abb. 4), ist fraglich: Trotz ausgefeilter numerischer Verfahren zur Lösung der zweidimensionalen Schrödingergleichung incl. transparenter Randbedingungen mussten Handoptimierungen für Matrixberechnungen und Speicherzugriffe implementiert werden, um halbwegs flüssige Animationen in den Java-Applets zu erzielen [11]. Für alle anderen bisher betrachteten Anwendungsgebiete sollte aber die Geschwindigkeit der Programme keine Rolle mehr spielen.

Der hohe Aufwand – bei reduziertem Nutzen gegenüber dem ursprünglichen Projekt – stand bisher einer kompletten Portierung der Applets entgegen. Man kann hoffen, dass automatische Java-nach-JavaScript-Compiler wie SwingJS zukünftig in der Lage sein werden, auch komplexe Swing-Anwendungen automatisch zu übertragen.

## Komplette Neuentwicklung in Matlab

Löst man sich immer weiter von den ursprünglichen Anforderungen und stellt die Frage, wie man physikalische Simulationen für die Lehre heute entwickeln würde, käme auch ein Werkzeug wie Matlab in Frage. Es stellt eine große Vielfalt komplexer numerischer Funktionen und ausgefeilter graphischer Möglichkeiten zur Verfügung bis hin zur Werkzeug-unterstützten Entwicklung graphischer Benutzeroberflächen. Es ist zwar proprietär, aber weit verbreitet, insbesondere auch in der Lehre. Alternativ könnte man auf Open-Source-Projekte wie Scilab [20] oder Octave [21] zurückgreifen, die beide Matlab ähnlich sind und einen erstaunlichen Funktionsumfang bieten.

Wie stark sich die Entwicklung damit gegenüber einer Java-Programmierung vereinfacht, zeigt die Beispiel-Simulation "Schwingerkette (s. Abb. 7). Ihr gesamter Code umfasst lediglich 263 Zeilen, dazu kommt eine Binärdatei, die die Informationen zum Aufbau der Benutzeroberfläche enthält und vom GUI-Builder automatisch erstellt worden ist. Das Programm stützt sich dabei wesentlich auf Matlabs Solver zur Lösung von Differentialgleichungen sowie die komplexen Plot-Routinen.

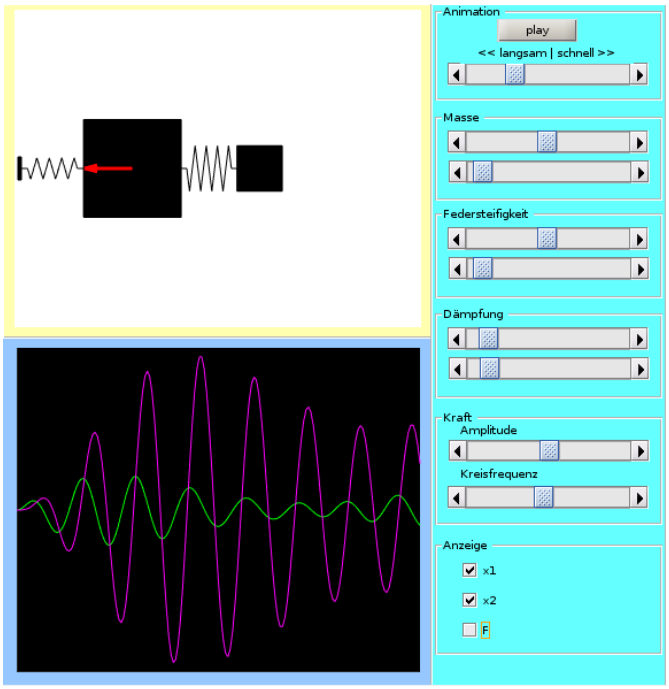


Abb. 7: Matlab-Version des Applets "Schwingerkette (Abb. 1)

Tatsächlich setzt der Autor schon lange Matlab als Prototyping-Werkzeug ein, bevor er die mühsamere Implementierung in Java beginnt.

Damit wurden nicht nur die komplexen numerischen Algorithmen der Wellen- und Quantenmechanik-Applets entwickelt, sondern auch Beispielergebnisse berechnet, mit denen die Applet-Resultate getestet wurden. Aufgrund von Matlabs optimierten Numerik-Funktionen war dabei die Performance sogar für die hohen Anforderungen der Quantenmechanik-Beispiele ohne weitere Optimierungen ausreichend.

Verglichen mit dem PhysBeans-Projekt bleibt dabei bisher eine wesentliche Anforderung offen: die direkte Einbindung in Webseiten, möglichst ohne eine Matlab-Installation beim Client. Für die Verbreitung fertiger Matlab-Applikationen ("Deployment") stellt Mathworks zwar diverse Werkzeuge zur Verfügung [22], die aber alle die hier gestellten Anforderungen verfehlen: Mit dem "Matlab Coder" kann ein C/C++-Programm erstellt werden, das aber keine graphischen Funktionen unterstützt. Mit dem "MATLAB Compiler SDK" werden Bibliotheken u. a. für Java und Python erzeugt, aber nicht für JavaScript. Und der "MATLAB Web App Server" erlaubt zwar die Verbreitung von Applikationen über das Internet, benötigt dazu aber einen dezidierten Server in einer streng geschützten Umgebung. Scilab bietet eine Cloud für Scilab-basierte Web-Anwendungen, aber ebenfalls keine clientseitigen Lösungen, Octave beinhaltet keinerlei Deployment-Unterstützung.

## Fazit

Damit die Simulationsprogramme des PhysBeans-Projekts trotz fehlender Browser-Unterstützung weiterhin funktionieren, werden sie auf der Projekt-Homepage über den Java-Web-Start-Mechanismus zur Verfügung gestellt. Wie oben beschrieben, ist dies aber leider nur eine temporäre Lösung.

Für größere Portierungsaktionen sind leider die Ressourcen des Autors nicht ausreichend – freiwillige Mitarbeit wäre hoch willkommen. Den geringsten Aufwand macht sicher die Portierung auf die libGDX-Plattform. Allerdings ist bei OpenSource-Projekten immer fraglich, wie lange die bisherigen Entwickler noch weiter machen bzw. ob sich neue finden. Auch die Stabilität der API ist nicht gewährleistet, so ließ sich das vor vier Jahren erstellte Beispiel bei einem Test vor zwei Jahren nicht mehr übersetzen.

Eine Neuentwicklung in JavaScript kommt momentan überhaupt nicht in Frage, es bleibt die Hoffnung auf bessere automatische Tools.

Vom Umfang her wäre eine Matlab-Version vielleicht sogar möglich, scheitert aber bislang an den fehlenden Deployment-Werkzeugen für Client-Anwendungen.

Seit kurzem ist allerdings ein weiterer Ansatz in Sicht, der die Portierung überflüssig machen könnte: Der WebAssembly-Standard [23] definiert ein Binärformat, das die Ausführung von Code nahezu mit der Geschwindigkeit kompilierter Programme in einem Browser ermöglicht. Er wird von den wichtigsten Browsern inzwischen unterstützt. Dazu kommen Compiler, die C/C++-Programme in das WebAssembly-Format übertragen. Geplante Weiterentwicklungen sollen die direkte Kompilation von Java ermöglichen. Damit könnte der vorhandene PhysBeans-Code auch in zukünftigen Webbrowsern verfügbar gemacht und das Projekt weitergeführt werden, statt die Energie auf Portierungen zu verschwenden.

# Literaturverzeichnis


[1] **Christian. W.; Belloni, M.**: *Physlet Physics.* Prentice Hall Upper Saddle River, USA, 1. Aufl. (2003).

[2] **Christian, W.; Esquembre, F.**: *Modeling Physics with Easy Java Simulations.* Physics Teacher, **45**, 475-480 (2007).

[3] **Junglas. P.**: *cliXX PhysBeans.* Verlag Harri Deutsch Frankfurt (2008).

[4] **Falstad, P.**: *Math and physics applets.* http://www.falstad.com/mathphysics.html/ .

[5] **Fendt, W.**: *Java-Applets zur Physik.* http://www.walter-fendt.de/ph14d/ .

[6] **Esquembre, F.**: *Easy Java Simulations Wiki.* http://www.um.es/fem/EjsWiki/Main/HomePage .

[7] **Christian, W.; Esquembre, F.**: *Physlet® Physics 3Ed.* http://www.compadre.org/books/Physlets-3E .

[8] **Hanson, R.**: *Why bother converting Java applets to JavaScript?* https://chemapps.stolaf.edu/swingjs/site/swingjs/examples/index.html .

[9] **Stöcker, H. (Hrsg.)**: *Taschenbuch der Physik mit CD-ROM.* Verlag Europa-Lehrmittel Haan-Gruiten, 7. Aufl. (2014).

[10] **Junglas, P.**: *Einsatz von Applets in der Physik-Ausbildung - Fallstudie Nichtlineare Systeme und Chaos.* Global J. of Engng. Educ., **7**, 337-347 (2003).

[11] **Junglas, P.**: *Transparent boundary conditions for simulation programs.* Wismarer Frege-Reihe, **3**, 60-65 (2010).

[12] **Junglas, P.**: *Homepage des PhysBeans-Projekts.* http://www.peter-junglas.de/fh/physbeans/index.html .

[13] **Zwintzscher, O.**: *Software-Komponenten im Überblick: Einführung, Klassifizierung & Vergleich von JavaBeans, EJB, COM+, .Net, CORBA, UML 2.* Springer Nature Campus Berlin (2004).

[14] **Oracle Corporation.**: *Overview of Java Web Start Technology.* https://docs.oracle.com/javase/10/deploy/overview.htm .

[15] **Oracle Corporation.**: *Java Client Road Map Update 2018-03-05.* http://www.oracle.com/technetwork/java/javase/javaclientroadmapupdate2018mar-4414431.pdf .

[16] **Zechner, M.**: *libGDX Cross-platform Game Development.* https://libgdx.badlogicgames.com/ .

[17] **Nair, S. B.; Oehlke, A.**: *Learning LibGDX Game Development.* Packt Publishing Birmingham, 2. Aufl. (2015).

[18] **Lawson, B.; Sharp, R.**: *Introducing HTML5.* Pearson Education San Francisco, 2. Aufl. (2011).

[19] **Loisel S.**: *Numeric Javascript.* http://numericjs.com/ .

[20] **scilab enterprises**: *Scilab – Open source software for numerical computation.* http://www.scilab.org/ .

[21] **Eaton, J. W.**: *GNU Octave Scientific Programming Language.* https://www.gnu.org/software/octave/ .

[22] **The MathWorks, Inc.**: *Mathworks – Desktop and Web Deployment.* https://de.mathworks.com/solutions/desktop-web-deployment.html .

[23] **Haas, A.; Rossberg, A.; et al.**: *Bringing the web up to speed with WebAssembly.* In: Proc. 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, New York, USA, 185-200 (2017).

## Autor

Prof. Dr. rer. nat. Peter Junglas

Private Hochschule für Wirtschaft und Technik Vechta/Diepholz

Schlesierstraße 13a

D-49356 Diepholz

E-Mail: peter@peter-junglas.de