Markus Gälli, Marcus Denker

Von kleinen und großen Erfindern

Squeak: Lernumgebung und Smalltalk-System für Kinder und Erwachsene

Es muss keine staubtrockene Angelegenheit von Informatikstudenten sein, dem Computer neue Dinge beizubringen. Squeak will Kindern nicht nur den Umgang mit dem Rechner näher bringen, sondern ihnen auch als Mittel zur Entdeckung der realen Welt dienen. Und für Erwachsene stellt es eine Smalltalk-Umgebung dar, die auch spielerisch erschlossen werden kann.

Die Zukunft sagt man am besten voraus, indem man sie erfindet", meint der Computer-Pionier Alan Kay. Eigentlich sollte er es wissen: Schließlich erfand er Ende der 60er-Jahre das Prinzip überlappender Fenster, entwarf erste Prototypen tragbarer Computer und prägte neben dem Begriff Objektorientierung auch den vom Personalcomputer. Kay ist überzeugt davon, dass die Nutzung des Mediums Computer auch heute noch in den Anfängen steckt. Die Multimedia-Lern- und -Programmierumgebung Squeak ist einer der Ansätze, die dies ändern sollen.

Die meisten Menschen, kritisiert Kay, simulieren mit dem Computer zurzeit andere Medien – der Computer ist Briefträger, CD-Spieler, Buchhalter oder Zeitung. Einen Computer zu programmieren dagegen wird ähnlich wie das Lösen mathematischer Aufgaben als trocken empfunden. Entsprechend wenige nehmen bislang diese Hürde.

Doch Programmieren muss keine Angelegenheit für verschrobene Computerfreaks sein, meint Kay. Ganz im Gegenteil: Computer bieten die Möglichkeit, komplexe Sachverhalte anschaulich zu simulieren. Wenn ein Kind am Computer ein Auto nicht nur malen, sondern auch festlegen kann, wie schnell es sich pro Sekunde vorwärts bewegen soll, lernt es am selbst geschaffenen Objekt etwas über Geschwindigkeit. Wenn es dann noch in jeder Sekunde die Geschwindigkeit erhöhen kann, lernt es auch etwas über Beschleunigung. Genau über solche Mechanismen soll Squeak Kindern nicht nur den Umgang mit Computern und ihre selbstständige Programmierung näher bringen, sondern auch selbst als Werkzeug zum Lernen einsetzbar sein.

Vorläufer

Alan Kay verfolgt seit mehr als 30 Jahren einen Traum: eine Maschine zu bauen, die den Möglichkeiten des Mediums Computer, „eines Instrumentes, dessen Musik Ideen sind", gerecht wird. Diese Maschine nennt er „Dynabook". Kinder jeden Alters sollten mit einem leichten, interaktiven und vernetzbaren Buch aktiv neue Inhalte schaffen und austauschen können – seien es nun wissenschaftliche Simulationen, multimediale Kunstwerke, interaktive Geschichten oder Spiele.

Wenn eine der Zielgruppen für solch eine Entwicklung Kinder sind, bringt das interessante Aspekte mit sich: Die Oberfläche muss nicht rückwärtskompatibel sein, dafür aber grafisch und somit leicht zu erlernen. „Jeder wusste 1970, wie sich die Chips entwickeln würden. Das Problem", erklärt Kay, „ war die Software." Nach Moores Gesetz sollte die Hardware ungefähr im Jahr 1980 so weit sein, also blieben zehn Jahre, die entsprechende kindgerechte Software zu programmieren. Damals wurde Kay eine Forschungsstelle am Forschungszentrum von Xerox (PARC) angeboten. Hier arbeitete er mit der „Learning Research Group" an diesem Problem.

Das Ergebnis war Smalltalk: Ein integriertes System aus objektorientierter Programmiersprache, Entwicklungsumgebung mit reichhaltigen Bibliotheken und grafischer Oberfläche. Es war die erste moderne grafische Bedienoberfläche mit überlappenden Fenstern, wie man sie heute auf jedem PC findet. Steve Jobs hat später die Oberfläche für den Apple-Computer „Lisa" komplett überarbeiten lassen, nachdem er bei Xerox eine Demonstration von Smalltalk gesehen hatte. Zu den berühmtesten Legenden der Computerbranche gehört, dass Jobs den Chefentwickler von Smalltalk, Dan Ingalls, fragte, ob man anstatt zeilenweise auch pixelweise scrollen könnte. Ingalls öffnete während der Vorführung einen Editor und änderte den ent-

sprechenden Code im laufenden System.

Für ihre Vision, Konzeption und Entwicklung des ersten vernetzten und brauchbaren Personalcomputers wurde den ehemaligen Kollegen vom Xerox PARC (Alan Kay, Butler Lampson, Robert Taylor und Charles Thacker) von der National Academy of Engineering Ende Februar 2004 der wichtigste Ingenieurspreis der USA, der Charles-Stark-Draper-Preis, verliehen.

Software-Spielzeug

Anfang der 80er-Jahre entwickelte sich Smalltalk immer mehr in Richtung auf eine professionelle Programmiersprache. Das 1983 vorgestellte und später kommerzialisierte Smalltalk 80 war eines der einflussreichsten Software-Projekte der vergangenen 30 Jahre, wurde aber dem ursprünglichen Ziel eines Computers für Kinder nicht gerecht. Mitte der 90er-Jahre nahmen Kay und seine Forschungsgruppe daher einen neuen Anlauf, die Ziele des Dynabook zu erreichen.

Die Squeak-Central genannte Gruppe startete bei Apple das Squeak-Projekt und veröffentlichte 1996 die erste Version unter einer freien und liberalen Lizenz im Internet. Später entwickelte Squeak-Central bei Walt Disney Imageniering die mit Squeak realisierten Etoys. Mit diesen basteln heute Schüler von Los Angeles [1] bis Kyoto [2] ihre ersten Simulationen oder Spiele. Kay gründete nach der Zeit bei Disney die Non-Profit-Organisation Viewpoints Research Institut [3], seit 2003 arbeitet er in der Forschungsabteilung von Hewlett-Packard weiter an der Zukunft des Lernens: „Wir haben damals die Windows-Idee eigentlich für Kinder entwickelt. Raten Sie mal, wer heute alles mit Windows arbeitet?“, kommentiert er die Entwicklung [4].

Den Umgang mit dem Computer und seine Programmierung bringt Squeak den Anwendern auf spielerische Weise nahe – gleichzeitig können vor allem Kindern auch Erkenntnisse über die Welt außerhalb des Rechners vermittelt werden. Aber auch Erwachsene finden nicht nur eine komplette Smalltalk-Umgebung, sondern können Simulationen beziehungsweise Spiele einfach zusammensetzen.

Informationen zu Squeak gedruckt, im Web und auf CD

Neben der Software findet sich im Web eine Fülle von Informationen zu Squeak – vieles in Englisch, aber einiges auch in deutscher Sprache (http://squeak.de/). Einige Squeak-Entwickler und -Nutzer haben 2002 den Verein Squeak Deutschland gegründet, um die weitere Entwicklung von Squeak zu fördern. Die Webseiten von Squeak Deutschland e.V. (http://squeak-ev.de) bieten weitere Informationen, sowohl über den Verein als auch über das System selbst.

Squeak e.V. hat zudem eine CD-ROM zusammengestellt, die neben der aktuellen Squeak-Version (3.6) auch eine Sammlung von Büchern und Tutorials rund um Squeak enthält. Das ISO-Image dieser CD ist über den Webserver des Vereins kostenlos verfügbar (http://squeak-ev.de/SqueakCD_de.html).

Besonders für Informationen zum Einsatz von Squeak in der Schule ist Squeakland (http://squeakland.org) erste Anlaufstelle. Die Site bietet weitere Hintergrundinformationen, Tutorials und Beispiele. Inzwischen ist auch ein thematisch passendes Buch erschienen [9], das Buch über Squeakland bestellt werden kann (http://squeakland.org/sqmedia/books/squeakerbooks.html). Auf Squeakland hat Alan Kay auch seine Literaturliste (http://squeakland.org/sqmedia/books/book_list.html) veröffentlicht, die er für seine Studenten als Hintergrundmaterial für die Ideen und Philosophien zusammengestellt hat, die die Entwicklung von Squeak beeinflussten.

Studenten der Ball State University haben einen preisgekrönten Dokumentarfilm (http://squeakersfilm.org/) über das Squeak-Projekt erstellt. Der Film ist über die Website als DVD erhältlich.

Erster Anlaufpunkt für Entwickler im Web ist http://squeak.org. Hier finden sich Links zum Squeak-Wiki (http://minnow.cc.gatech.edu/squeak) und zu der Squeak-Entwicklermailingliste. Zum Thema Smalltalk allgemein bieten www.whysmalltalk.com und die Website der deutschen Smalltalk-Usergroup (www.gsug.org) weitere Informationen. Eine gutes englischsprachiges Tutorial über Smalltalk mit Squeak gibt es unter www.cosc.canterbury.ac.nz/~wolfgang/cosc205/smalltalk1.html.

Squeak ist ebenso freier, multimedialer Experimentierkasten wie eine professionelle Entwicklungsumgebung auf Smalltalk-Basis.

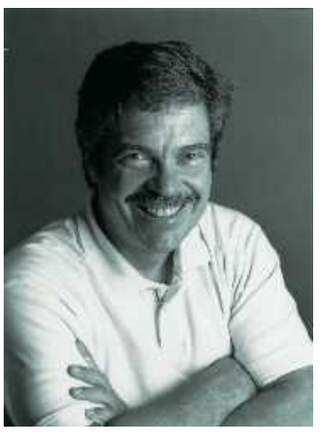
Es gibt zwei unterschiedliche Squeak-Varianten: Das Squeakland-Browser-Plug-in, eine abgespeckte Squeak-Version, die für das folgende Beispiel am geeignetsten erscheint. Das Plug-in kann man sich über www.squeakland.org im Internet-Browser installieren. Sehr hilfreich für den Einsteiger ist auch die Kurzanleitung „Quickstart-Guide“, die es unter www.squeakland.org/author/startguide.html gibt. Leider ist diese bisher nur auf Englisch verfügbar. Wer tiefer in die Welt der Smalltalk-Programmierung einsteigen will, dem sei die vollständige Squeak-Entwicklungsumgebung von der CD der c’t-Ausgabe 20/03 oder von http://people.squeakfoundation.org/ empfohlen.

Als Beispiel für den Einsatz der grafischen Entwicklung sollen ein Auto und eine Rennpiste dienen: Das Auto lässt sich zum einem selbst mit einem Lenkrad steuern; mit Hilfe von Sensoren kann es aber auch automatisch über die Piste manövrieren. Mit mehreren solcher selbstgestalteter Autos lassen sich auch Rennen veranstalten.

Fahrn, fahrn, fahrn

Zur Erzeugung der kleinen Auto-Rennwelt malt man mit Squeak einfach ihre Elemente. Dazu öffnet man nach dem Start des Systems die orange Lasche („Flap") namens „Navigator" und mit dem Pinsel die Malpalette (siehe Abbildung unten). Nach der Gestaltung des Autos übernimmt man es mit dem Befehl „keep" dauerhaft ins System. Das Auto lässt sich mit einem Klick der linken Maustaste anheben und beliebig verschieben, ein weiterer Klick lässt es wieder fallen. Das Auto wirft nach dem Anheben einen Schatten: Es ist nicht etwa ein rechteckiges Bild, sondern ein freistehendes Symbol.

Ein Linksklick auf das Auto bei gedrückter Alt-Taste lässt die „Halos" des Auto-Objekts erscheinen. Die so genannten



Halos dienen zu komplexeren Manipulationen eines Objekts, die über einfache menügesteuerte Aktionen hinausgehen. So lässt sich ein Objekt – abhängig von seinen Möglichkeiten – über seine Halos etwa gezielt verschieben, kopieren, vergrößern und so weiter. Bei Anzeige der Halos erscheint außerdem der

Zu dem innerhalb von Squeak erstellten Auto gehören bestimmte „Halos", die zu komplexeren Manipulationen eines Objekts dienen. Das Auto-Objekt zeigt zudem sein Rotationszentrum.

Name des Objekts, der sich nach Linksklick auf den Text in „Auto" ändern lässt. Analog zum Auto lässt sich eine Rennpiste bauen; das Auto wird nun einfach auf die Piste gesetzt.

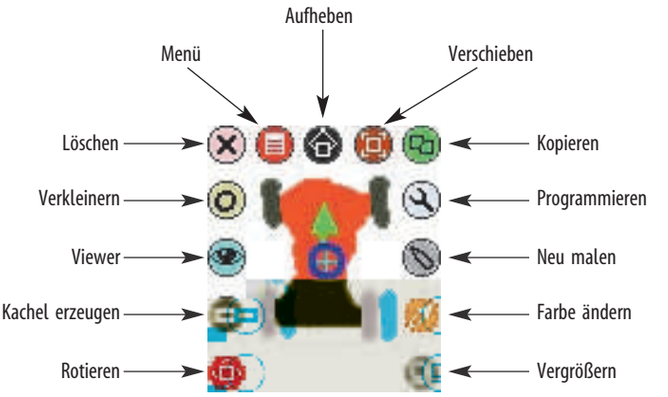
Über den Halo „Viewer" erreicht man eine Leiste mit den für dieses Auto verfügbaren Befehlen (siehe Abbildung auf S. 220 oben). Dazu gehören etwa die Befehle „Auto forward by 5" und „Auto turn by 5". Mittels Drag & Drop lassen sich diese Befehle auf die Oberfläche ziehen; dadurch entstehen zwei neue Skripte. Durch einen Klick mit der linken Maustaste auf das Ausrufezeichen lassen sich die Skripte ausprobieren: Das Auto bewegt sich um fünf Schritte nach vorne beziehungsweise dreht sich um fünf Grad im Uhrzeigersinn.

Falls sich das Auto allerdings seitlich bewegt, war seine Aus-

Der Computer-Pionier Alan Curtis Kay verfolgt seit mehr als 30 Jahren den Traum, eine Maschine zu bauen, die den Möglichkeiten des Mediums Computer wirklich gerecht wird.

gangsrichtung falsch eingestellt: Jedes Objekt hat eine interne Richtung, anfangs „zwölf Uhr". Das Rotationszentrum eines neuen Objektes befindet sich in der Mitte. Das Zentrum kann aber visuell verändert werden, indem man im Menü des Autos „Direction Handles" auswählt und anschließend mittels Alt-Klick erneut die Halos des Objektes anzeigt. Der grüne Pfeil kann nun in die Vorzugsrichtung des Objektes gedreht werden.

Nach dem Öffnen der Palette in Squeak lässt sich einfach ein Auto malen und mit „keep" übernehmen.



Auto

Wann die Skripte ausgeführt werden sollen, lässt sich wiederum über ein Popup-Menü einstellen. Mit der Einstellung „ticking" wird ein Skript jede Sekunde angestoßen; „paused" hält die Skripte vorübergehend an. Um alle Skripte innerhalb des Projekts auf einmal zu starten, zu beenden oder schrittweise auszuführen, existiert ein eigenes Werkzeug namens „All Scripts", das sich in der Ablage „Widgets" befindet. Startet man alle Skripte über dieses Werkzeug, fährt das Auto im Kreis.

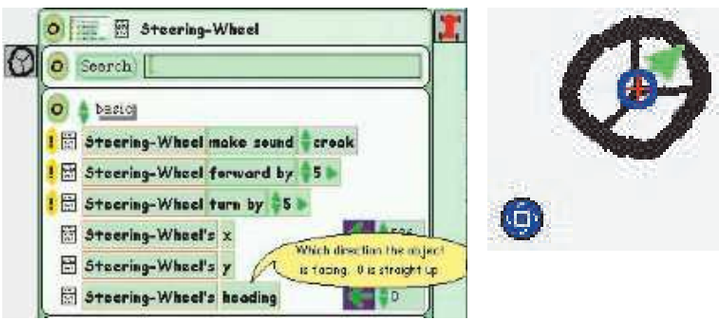
Rennfahrer

Eine Manipulation des Autos über solche Skripte ließe allerdings nur relativ unflexible Fortbewegung zu. Zur manuellen Steuerung soll daher ein Lenkrad eingesetzt werden – dieses wird wieder mit dem Malwerkzeug gestaltet. Die Steuerung des Autos mit dem Lenkrad bedeutet, es um soviel Grad zu drehen, wie das Lenkrad eingeschlagen wird (siehe Abbildung auf S. 220 mitte). Nach dem Öffnen der Befehlsleiste des Lenkrads zieht man daher mittels Drag&Drop die Richtung des Lenkrads auf die Gradangabe des Skripts, mit dem sich das Auto rotieren ließ.

Nach dem Neustart der Skripte kann man den Versuch starten, das Auto auf der Piste zu halten, indem man das Lenkrad dreht – in diesem Stadium kein einfaches Unterfangen. Daher soll eine inverse Servolenkung Abhilfe schaffen: Die Idee ist, das Auto nur um einen Bruchteil der Rotation des Lenkrads zu drehen. Dazu klickt man auf das Dreieck rechts von „Lenkrad-Richtung" („Steering Wheels' heading"). Die Kachel erweitert sich um „+1". Nach einem Klick auf das „+" wählt man aus dem aufgehenden Menü „/" aus, die „1" erhöht man mit den Pfeilen auf „5" (siehe Abbildung auf S. 220 mitte).

Um nun eine automatische Steuerung zu entwickeln, die das Auto selbstständig auf der Piste hält, stoppt man zunächst die Skripte und dreht das Lenkrad auf seine Grundposition. Für die Steuerungsautomatik kann man dem Auto Sensoren verpassen; es soll dann nach rechts gegensteuern, wenn sein linker Sensor eine Farbe erkennt, die nicht der Pistenfarbe entspricht. Analog soll das Objekt nach links steuern, wenn sich sein rechter Sensor nicht mehr auf der Piste befindet. Um dies zu realisieren, benötigt man zunächst wieder das Malwerkzeug des Autos, das man über das Halo „Neumalen" erreicht. Damit lässt sich an der linken Seite des Autos ein grüner, rechts ein blauer Sensor erstellen.

Anschließend zieht man den Rotationsbefehl des Autos aus der Befehlsleiste auf die Oberfläche und erzeugt dadurch ein neues Skript. Damit soll programmiert werden, dass sich das Auto nach rechts – also im Uhrzeigersinn – dreht, wenn der linke grüne Sensor eine Farbe sieht, die nicht die Pistenfarbe ist. Dazu zieht man das gelbe „Test-Wahr-Falsch-Kachel" in das Skript (siehe Abbildung auf S. 220 unten). Als Bedingung soll gelten, dass die grüne Farbe des Sensors das Orange der Rennpiste sieht. Dies lässt sich realisie-



ren, indem man in der Befehlsleiste des Autos die Kategorie „Tests“ auswählt und dort die Kachel „Auto’s color sees“ in die Testzeile des Skripts bewegt. Die Farben stellt man ein, indem man erst auf die farbig ausgefüllten Rechtecke der eben erzeugten Kachel mit Namen „Auto’s color sees“ klickt und über die dadurch auftauchende Pipette die jeweiligen Farben von Auto und Rennpiste abgreift. Nun muss noch der Rotationsbefehl im Skript in die „No-Zeile“ verschoben werden – schließlich soll sich das Auto nur dann drehen, wenn es nicht auf der Piste steht. Analog erstellt man das Skript für die Rotation des Autos nach links.

Nach dem Start der Skripte sollte das Auto nun selbstständig über die Piste kurven. Kopiert man das Auto mit dem „Kopieren-Halo“, lassen sich die Auto-Objekte unterschiedlich programmieren und gegeneinander antreten – eine kleine Rennsimulation ist entstanden.

Die Squeak-Umgebung ermöglicht natürlich nicht nur die Programmierung von Autorennen. Ältere Kinder können mit Squeak komplexe Projekte verwirklichen – ganz alte Kinder sozusagen haben ebenfalls ihre Freude an solchen Projekten und entwickeln Computerspiele: Alan Kays Lunar Lander (siehe Abbildung auf S. 221 oben) beispielsweise, der das Landen einer Mondfähre zum Ziel hat, ist ein einfaches Videospiel. Allerdings muss man sich, wenn man es mit Squeak selbst erstellen möchte, einiges an Wissen aneignen, das nichts mit der Programmierung des Computers zu tun hat: Ohne physikalische Kenntnisse über Geschwindigkeit, Beschleunigung, Fallgesetze und so weiter gelingt dies nicht.

Objektorientiert

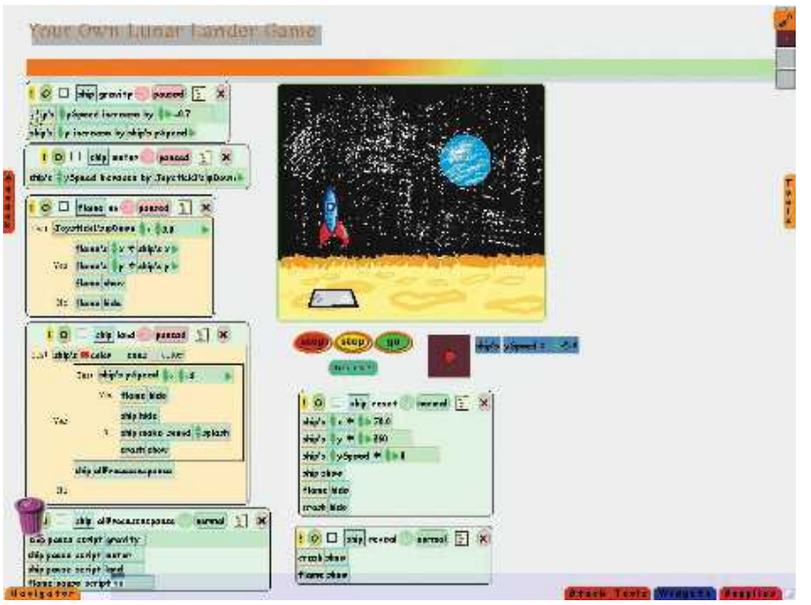
Squeak ist aber nicht nur ein Spielzeug und Lernmittel für Kinder und ein System zum Bau mehr oder weniger komplexer Spiele für Erwachsene, sondern auch eine komplette Smalltalk-Umgebung. Und in noch einer Hinsicht ist es auch für Experten interessant: Squeak ist (fast) komplett in Squeak selbst implementiert, alles ist für Experimente und Erweiterungen zugänglich. Es eignet sich also hervorragend für die Erstellung von Prototypen und zum Experimentieren. Für den Profi ist Squeak kein Spielzeug, sondern bietet eine mächtige Entwicklungsumgebung: Neben den Etoys bietet es ein Programmiersystem mit Smalltalk als professioneller Sprache.

Squeak ist frei verfügbar, aller Quellcode steht immer und sofort zur Verfügung. Die internationale Squeak-Entwicklergemeinde ist seit der ersten Veröffentlichung 1996 rasch gewachsen. Smalltalk als objektorientierte Sprache macht das System als Umgebung für nor-

male Programmierprojekte interessant. Da Squeaks virtuelle Maschine in Squeak selbst programmiert wurde, sind die in Squeak realisierten Systeme einfach portierbar. Die virtuelle Maschine bietet zudem einen Garbage Collector, der Programmierer muss sich also nicht mehr um die Details der Speicherverwaltung kümmern. Squeak ist dabei zurzeit auf über 20 Plattformen verfügbar – vom PDA bis zum Server.

Squeak ist vollständig objektorientiert. Zahlen, Klassen, Methoden, auch die Stackframes sind Objekte. Die Klassenbibliothek trägt mit einer Bandbreite von grundlegenden Funktionen (beispielsweise Collection Classes) bis hin zu 3D-Grafik und Multimedia in der Regel allen Entwicklungsansprüchen Rechnung. Der „Edit-Compile-Run"-Zyklus ist dabei sehr klein: Jede geänderte Methode wird sofort nach dem Speichern übersetzt und ist direkt ausführbar. Dies ermöglicht, auch zur Laufzeit eines Programms Fehler zu beheben, ohne das Programm zu beenden.

Die Programmiersprache selbst ist dabei recht einfach und versperrt nicht den Blick auf das Wesentliche: Nicht ohne Grund sind wichtige Entwicklungen im Software-Engineering im Smalltalk-Umfeld entstanden, etwa das Konzept des Model-View-Controller (MVC), Extreme Programming, Test Driven Development oder Design Patterns.



Alan Kay's Lunar Lander: Das Ziel dieses klassischen Spiels ist das Landen einer Mondfähre.

Squeak ist nicht nur Spielzeug für Kinder, sondern bietet eine mächtige Entwicklungsumgebung mit Smalltalk als professioneller Sprache.

Real Life

Ein Beispiel für den Einsatz von Squeak das Seaside Web Framework [5]. Seaside verfolgt das Ziel, die Entwicklung von Webanwendungen zu vereinfachen. Die Struktur einer Seaside-Applikation ist der einer normalen Anwendung mit grafischer Bedienoberfläche sehr ähnlich: Eine komplette User-Session kann als ein kontinuierliches Stück Quellcode mit natürlichem linearen Kontrollfluss implementiert werden; Webseiten können sich gegenseitig aufrufen wie Subroutinen. Dafür sind im Zusammenhang mit dem Projekt diverse Bibliotheken und Werkzeuge für größere Projekte entstanden, darunter beispielsweise Datenbankanbindungen. Außerdem existiert ein mächtiges Quelltext-Versionierungssystem mit Anbindung an CVS.

Auch am Medialab des MIT [6], am Institut für Computerspiele der Universität von Magdeburg [7] und bei der Software Composition Group in Bern [8] wird mit Squeak gelehrt und geforscht. An der Universität der Künste in Berlin hält Alan Kay eine Ehrenprofessur inne, auch dort finden regelmäßig Squeak-Kurse statt. Eines der wichtigsten Ziele der aktuellen Entwicklerversion ist die Internationalisierung: Squeak soll wie die Etoys-Oberfläche unter anderem komplett auf Japanisch, Spanisch und auch Deutsch verfügbar werden. (jk)

Literatur

[1] Open Charter School: www.squeakland.org/learn/elementary.html

[2] Schülerprojekt in Kyoto: www.squeakland.org/images/news/news.htm#embrace

[3] Alan Kays Viewpoints Research: www.viewpointsresearch.org

[4] Interview mit Alan Kay: http://zeus.zeit.de/text/2002/47/P-Alan_Kay

[5] Seaside Web Framework: www.beta4.com/seaside2/

[6] Squeak am MIT: http://llk.media.mit.edu/projects/summaries/scratch.shtml

[7] Wie mache ich ein Computerspiel: http://isgwww.cs.uni-magdeburg.de/games/start/kinder.html

[8] Software Composition Group: www.iam.unibe.ch/~schaerli/smalltalk/traits/traitsPrototype.htm

[9] B. J. Allen-Conn, Kim Rose, Powerful Ideas in the Classroom; Using Squeak to Enhance Math and Science Learning, 2003, viewpointsresearch.org, Bestellung über www.squeakland.org

[10] Mark Guzdial, Squeak: Object-Oriented Design with Multimedia Applications, 2001, Prentice Hall, ISBN 0-13-028028-3

[11] Mark Guzdial, Kim Rose, Squeak: Open Personal Computing and Multimedia, 2002, Prentice Hall, ISBN 0-13-060812-2

c't