SQUEAK
Informatik in einer schönen Smalltalk-Umgebung
Ein Tutorial für den Einsatz in der Schule

Heiko Schröder

1. März 2006

New Release:

eToys (Squeak 3) »fired out 28.02.06« 

Next Releases:

StarSqueak oder Kedama (Squeak 8) Frühjahr 2006
Planung Smalltalk III: Turtle-Grafik (Squeak 7)

Squeak 1: Eine Reise in eine bunte Welt mehr dazu siehe: 3.1
Squeak 2: Squeak-Grundlagen mehr dazu siehe: 3.2
Squeak 3: eToys mehr dazu siehe: 3.3
Squeak 4: Smalltalk-Hacking mehr dazu siehe: 3.4

Woran wird gerade gearbeitet? (PDF Snapshot, soweit vorhanden)

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Meinen tapferen Schüler der Deutschen Schule Prag.

Möge dieses Tutorial den Erfindern und den Entwicklern von Squeak den größten Dank aussprechen
und den Schulen dabei behilflich sein, die Schleife zu ziehen,
um dieses schöne Geschenk zu öffnen.

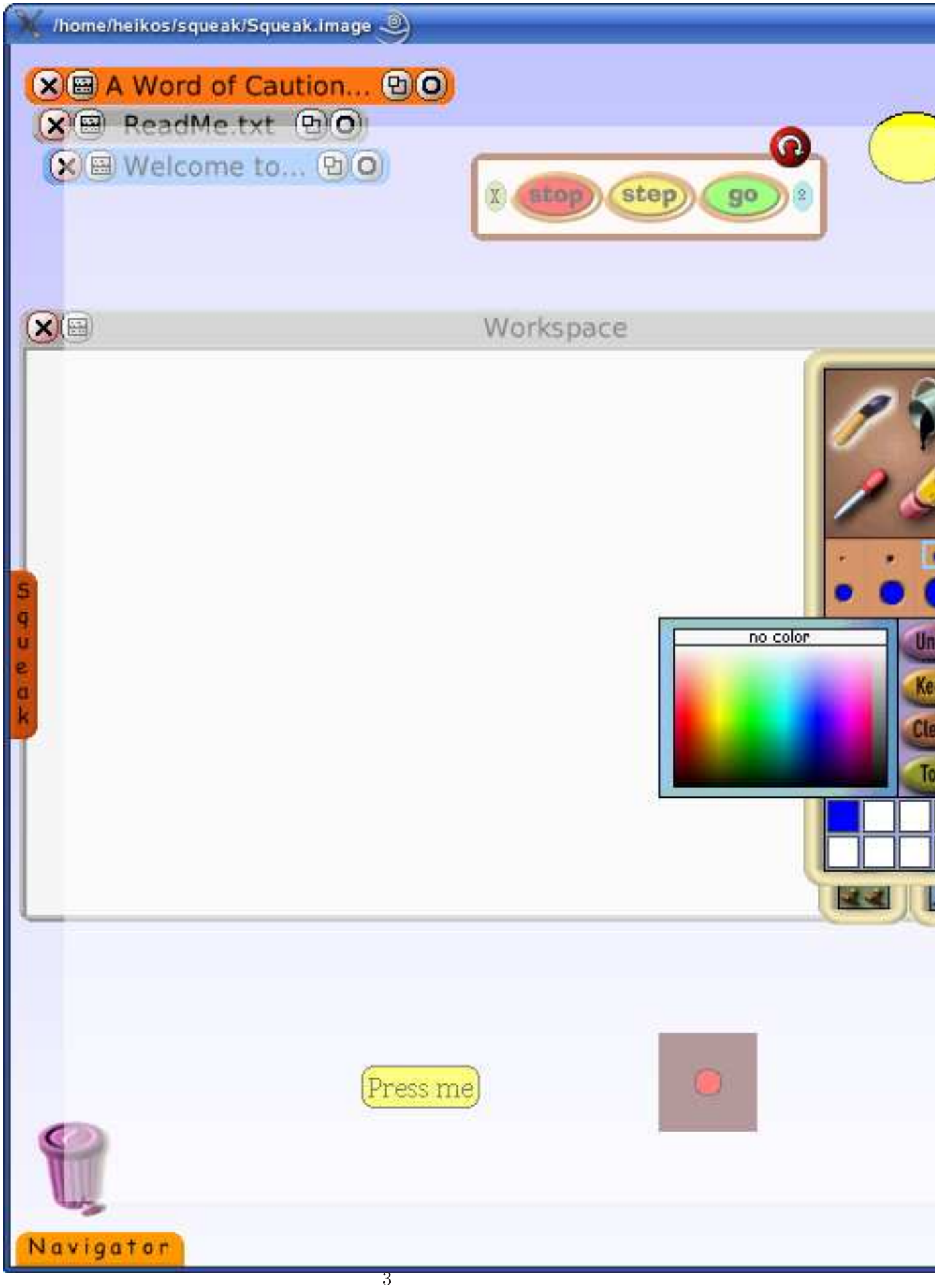


Abbildung 1: Der »Urwald« von Squeak

Squeak ist eine Umgebung, die ganz wesentlich auf der Programmiersprache SMALLTALK basiert. Genauer gesagt: es ist der Standard Smalltalk-80. Sicherlich haben einige der Leser schon von dieser Sprache gehört, aber bisher noch nie eine Smalltalk-Umgebung gesehen.

Die Abbildung 1 gibt einen ersten Eindruck. Einigen mag dieser Eindruck etwas »verspielt« erscheinen. Doch das ist ein sehr großer Irrtum. Squeak richtet sich zwar ganz ausdrücklich an Kinder. Aber an echte Profis und solche, die es werden wollen, gleichermaßen. Einen Smalltalk-Schreibtisch zeigt das Bild 2 . Squeak bietet neben den üblichen Werkzeugen (Workspace, Transcript und System-Browser) noch weitere, extrem leistungsfähige Werkzeuge, die über die Fähigkeiten manche anderer Smalltalk-Umgebungen hinausgehen. Dieses wird vor allem Schüler der oberen Mittelstufe, der Oberstufe und – wie gesagt – professionelle Programmierer interessieren.

Die jüngeren, angefangen mit der Grundschule, werden durch eine starke Erweiterung der so genannten Turtle-Grafik (Programmiersprache Logo) behutsam in die Welt des Programmierens eingeführt. Durch die großartige Idee, Programme in Form von Skripten durch das Ziehen von Kacheln zusammenzubauen, können sich die Kleinen ganz auf das Denken konzentrieren. Denn die Kacheln selbst sind syntaktisch richtig. Einen ersten Eindruck davon, wie diese Abteilung von Squeak aussieht, die sich eToys nennt, zeigt die Abbildung 3 .

In der oberen linken Ecke ist eine abermalige Erweiterung der Turtle-Grafik zu erkennen: StarSqueak, eine Erfindung von Michael Resnick (StarLogo). Hier werden Hunderte oder Tausende von Turtles gesteuert, mit denen biologische Simulationen durchgeführt werden können. Das kann die Frage nach der Formation von Zugvögeln, der Enstehung von Viruserkrankungen, die Ausbreitung eines Waldbrandes oder – nicht biologisch – die Entstehung von Verkehrsstaus sein.

Smalltalk wurde Anfang der siebziger Jahre von Alan Kay, Dan Ingalls, Adele Goldberg und ihren Mitarbeitern am Pala Alto Research Center (PARC) in Kalifornien erfunden. Viele dieser Erfinder, einschließlich Alan Kay sind nicht nur Informatiker, sondern (Mikro)Biologen. Angeregt von der norwegischen Vorläufersprache Simula entwickelten sie die Idee, Datensätze wie biologische Lebewesen aufzufassen, weiter. Der Computer arbeitet als Maschine nicht *mit* diesen Objekten, sondern die Objekte reagieren *selbst* wie Lebewesen auf Nachrichten (Reize), die ihnen andere Objekte oder der Programmierer senden. Dies war die Geburt der *objektorientierten Programmierung*.

Von Beginn an gehörte eine grafische Umgebung als »Welt« für diese Lebewesen zu der Sprache dazu. In den Zeiten der kleinen Arbeitsspeicher stellte diese Anforderung natürlich ein recht großes Problem dar. Deshalb bekamen Schulen und private Programmierer allein aus Konstengründen kaum eine Smalltalk-Umgebung zu Gesicht. Die großen Entdeckungen führten aber zur Entwicklung der grafischen Betriebssysteme, die alle ohne Smalltalk nicht denkbar wären. Glückliche Umstände unterstützten diese Entwicklung. Seymour Papert hatte am MIT mit der Turtle-Grafik nicht nur eine wesentliche Vorarbeit für die Entwicklung der objektorientierten Programmierung geleistet, sondern gleich eine Antwort auf eine Kernfrage Alan Kays gegeben: »Was werden Kinder mit einem PC anfangen können«? Smalltalk ist also nicht nur diejenige Programmiersprache, die den engsten Bezug zur Natur aufweist, sondern auch diejenige, bei denen Entwicklung von Beginn an Kinder gedacht wurde. Der zweite glückliche Umstand war die Entwicklung der Computer-Maus durch Douglas Engelbart.

Die Erfolge von Smalltalk führten 1983 zu einer angeblichen »Erweiterung« der Programmiersprache C in C++ durch Dr. Bjarne Stoustrup. Tatsächlich entstand eine völlig eigenständige Sprache, die aber es nicht wagte, Smalltalk-Ideen in ganzer Konsequenz umzusetzen. Damit ist C++ eine – allerdings sehr leistungsfähige – Hybridsprache geworden.

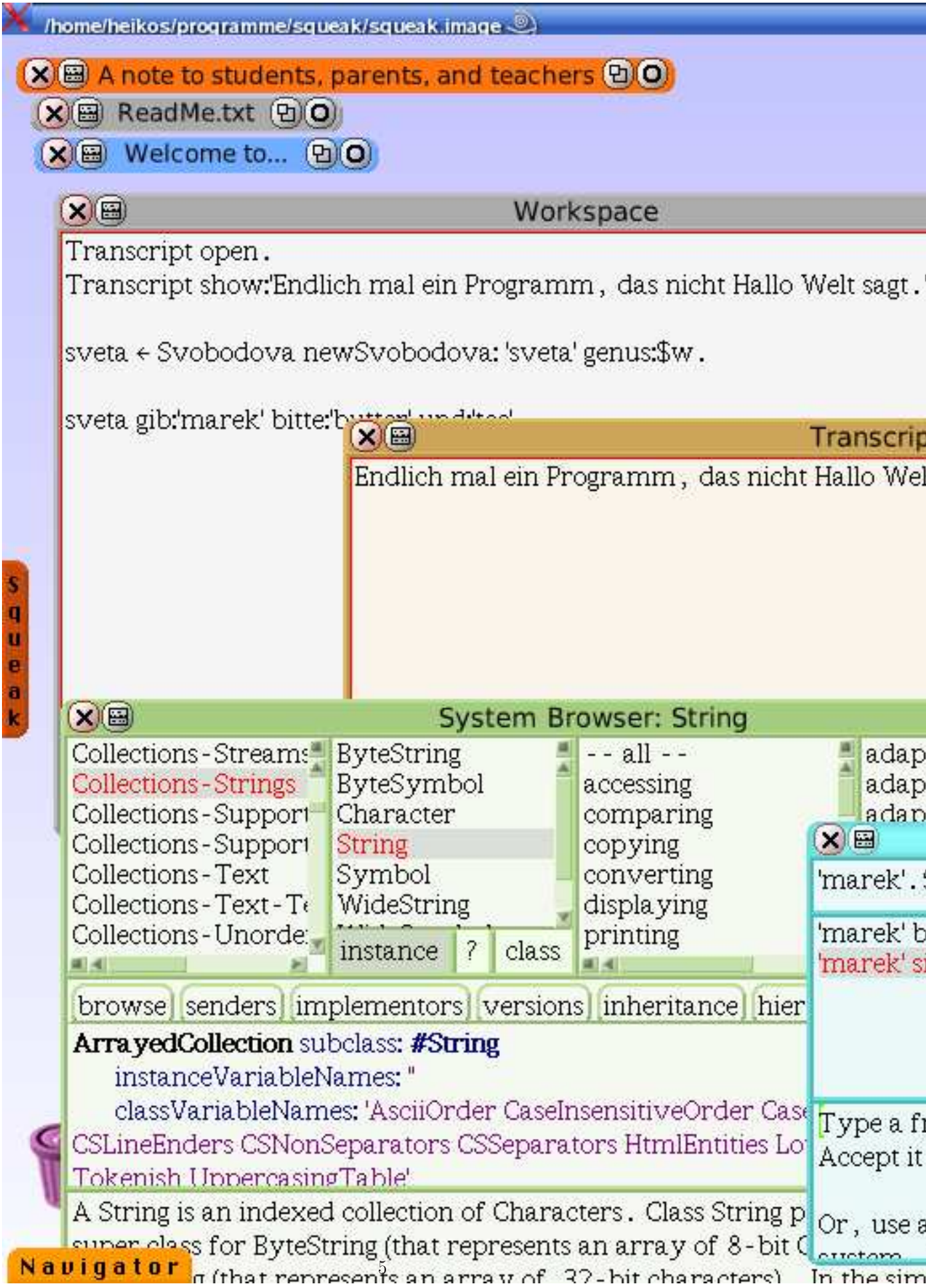


Abbildung 2: Smalltalk-Schreibtisch

Abbildung 3: Es »turtlet« in Squeak an allen Ecken.

Sehr viel konsequenter ging die Entwicklung von Java vor sich. Java ist in sehr vielen Details pures Smalltalk. Oder besser gesagt: Java folgt einem Smalltalk-Ableger mit dem Namen *Self*. Da die grafische »Welt« von Smalltalk in einer hardwareunabhängigen Image-Datei gespeichert ist, die von einer so genannten virtual machine (VM) lediglich interpretiert werden muss, können wir diese Image-Datei als erste Form eines Java-Applets auffassen. Denn Java arbeitet nicht anders. Die VM ist – entgegen mancher Ansichten – keine Java-Erfindung. Sie gab es schon vor Smalltalk. Leider orientiert sich Java allein aus Marketinggründen an der Syntax von C++ und nicht von Smalltalk.

Während dieser Jahre hat Alan Kay seinen Traum von einer Programmierumgebung für »Kinder jeden Alters« nie aufgegeben. Nach seinem Fortgang von PARC setzte Adele Goldberg die begonnene Arbeit fort, die zu dem definitiven Standard Smalltalk-80 führte. Inzwischen entwickelte Kay nach der sehr erfolgreichen Zeit bei Atari ein Projekt mit dem Namen Vivarium bei Apple. Dieses Vivarium war eine biologische Simulation eines Lebensraums, wobei der Smalltalk-Dialekt Smalltalk/V entstand. Nach dem Ende dieses Projektes fand sich die Gruppe, die einst am PARC Smalltalk entwickelte bei Apple wieder und der alte Traum wurde wieder wach.

In unglaublich kurzer Zeit entstand Squeak. Der alte Traum von einem Computer für »Kinder jeden Alters« wurde nicht nur Wirklichkeit, sondern die Entwicklung ging – dank der Erfahrungen mit dem Vivarium-Projekt – weit darüber hinaus. Squeak wurde eine »Spielwiese«, Turtle-Welt, Multimedia-Programmierumgebung und Smalltalk-Umgebung gleichermaßen. Kurz gesagt: Squeak ist ein endloses Abenteuer des Entdeckens, Forschens, Weiterentwickelns.

Der Wunsch nach Spaß in der Schule ist verständlich. Aber es ist nach Alan Kay sehr mißverständlich, welcher Spaß hier gemeint ist. Es ist nicht die »flache Freude«, sondern der »hard fun«, der in Wirklichkeit von Kindern erwartet wird. Ein Spielzeug, was nicht interessiert, landet irgendwo in der Ecke. Squeak ist aber immer wieder eine Herausforderung. Allein die *Möglichkeit*, 3D zu Programmieren, sich Internetwerkzeuge selbst zu entwickeln oder gar – für die ganz harten – die Veränderung der VM zur Entwicklung einer *eigenen* Programmiersprache (das ist möglich weil die VM *ebenfalls* in der Smalltalk/Squeak entwickelt ist), das läßt die *wirklich* Interessierten nicht so schnell los[1].

Dieses Tutorial will den Einstieg erleichtern. Da in Squeak *für* Kinder *jeden* Alters geschaffen wurde, und damit besonders für Schulen, handelt es sich als OpenSource Produkt um ein unglaubliches Geschenk was hier der Lehre überreicht wird. Die Beschäftigung damit dürfte seine Wirkung nicht nur in der Informatik entfalten, sondern vor allem auch in denjenigen Bereichen, die zu einer Teilnahme an Jugend forscht führen können.

Möge dieses Tutorial ein inneres »Feuer« entfachen.
Heiko Schröder, Prag, im Februar 2006

1.1 An wen richtet sich das Tutorial? Copyleft, Verantwortlichkeit

Das Tutorial richtet sich in erster Linie an Schüler ab der oberen Mittelstufe – wenn das Tutorial als *Ganzes* betrachtet wird. Einzelne Teile daraus können auch von jüngeren Schülern verstanden werden. Im engeren Sinne »gehört« dieses Tutorial zur Deutschen Schule Prag und im Besonderen der Klasse 10b des Jahres 2006. Es richtet sich also vor allem an Schüler, die Deutsch nicht als Muttersprache sprechen, diese Sprache aber seit der sechsten Klasse lernen.

[1] Alan Kay und seine Mitarbeiter forschen selbst bereits weiter und haben das Netzprojekt Croquet gestartet (siehe www.croquet.org), das auf Squeak aufsetzt.

Alle Teile dieses Tutorials unterliegen dem Free Documentation License der Free Software Foundation. Einzelheiten müssen unter www.gnu.orgnachgelesen werden. Sämtliche Links von dieser Seite, sowie allen Teilen des Tutorials, führen zu offiziellen Instituten. Für die Inhalte dieser verlinkten Seiten ist niemand der Autoren dieses Tutorials verantwortlich. Ferner sind die downloadbaren Teile ein Service, aus dem keinerlei Regressansprüche bei Fehlern technischer Art oder welcher Art auch immer entstehen. Der Leser nutzt dieses Angebot auf eigene Gefahr! Es wurde aber durch die Bereitstellung auf dem Server eines nicht öffentlichen Anbieters (Deutsches Forschungsnetz www.od.shuttle.de auf bestmögliche Absicherung gegen Viren oder anderer krimineller Angriffe geachtet, wenngleich eine Garantie nicht übernommen werden kann.

1.2 Überlegungen zur Sprache und zur Informatik

Programmieren hat mit Informatik genau so viel gemein wie der Rechenknecht vergangener Tage mit dem Mathematiker. Aus diesem Grunde geht es hier nicht darum, für die Sprache Smalltalk / Squeak zu werben. Wer weiterhin C++, Java oder was auch immer, im Unterricht einsetzen möchte, soll es tun. Viele Lehrpläne schreiben – glücklicherweise – keine Programmiersprache mehr vor. Für Begegnungsschulen treten aber gewisse Zwänge in den Vordergrund.

Alltagssprache: An Begegnungsschulen kann eine wohlüberlegte erste Programmiersprache den Schülern *sehr* helfen. Es ist zu bedenken, dass der Unterricht mit Deutsch als einer *Fremdsprache* verfolgt werden muss. Daher bedeutet eine Programmiersprache, deren Syntax sich an der Alltagssprache orientiert, einen enormen Vorteil. Ein C++, Java oder Python-Konstrukt wie

```
sveta.gibBitteUnd('marek','dieButter','denTee')
```

ist sicherlich fragwürdiger als Smalltalks Version

```
sveta gib:'marek' bitte:'dieButter' und:'denTee'.
```

Smalltalk lehrt damit zugleich – sehr subtil – das Bilden ganzer Sätze, was von Schülern beim mündlichen Gespräch aus Angst vor Fehlern gerne vermieden wird. Aus diesem Grunde wird bei den Kerninhalten des Tutorials zunächst die Sprache so weit wie notwendig dargestellt (Squeak 1 bis Squeak 4), während die restlichen Abschnitte (Squeak 5 bis Squeak 9) einen engeren Bezug zu sprachenunabhängigen Inhalten der Informatik herzustellen versuchen.

Konzept: Das zweite Kriterium, was in diesem Zusammenhang gar nicht hoch genug eingeschätzt werden kann, ist die Sicherstellung eines konsequenten Konzeptes. Von Vertretern anderer Meinungen wird dies gerne mit »Purismus« verwechselt. Ich wage die Behauptung, dass es unnötig schwierig ist, das objektorientierte Denken mit einer Hybridsprache zu lernen. Java ist – entgegen vieler Meinungen – nicht konsequent objektorientiert. Sie orientiert sich wesentlich an dem Smalltalk-Ableger *Self*, der ganz wesentlich mit so genannten *Prototypen* arbeitet. Der grafische Bereich von Squeak, der sich zum Beispiel mit eToys (allgemein mit Morphic) beschäftigt, verwendet ebenfalls *Self*. Der Squeaker wird dies allerdings als Marginalie betrachten, da das echte Self wenigstens die Syntax von Smalltalk verwendet. Dass die Kopie einer Ellipse eine *wirkliche* Kopie eines Prototyps darstellt und eigentlich mit der Instanziierung eines Objektes über eine Klasse nichts zu tun hat, wirkt sich nicht störend aus.

Java verwendet allerdings – trotz *Self-Konzepten* – die eher an prozeduralem Denken orientierte Syntax von C++. Weder Klassen, noch Anweisungsblöcke sind Objekte. Zahlen ebenfalls nicht. Zugriffsbeschränker wie public, protected oder Typendeklarationen bei Variablen haben in einer echten objektorientierten Sprache eigentlich nichts verloren (siehe Erläuterungen in Squeak 1). Python verzichtet denn auch teilweise darauf, benutzt aber ebenfalls eine an C++ orientierte Syntax. Das bedeutet eine unnötige weitere Schwierigkeit für Zweit- und Fremdsprachler, die an inländischen Schulen wahrscheinlich weniger ins Gewicht fällt.

Syntax: Das dritte Kriterium ist eine Syntax, die auf alle unnötigen Fehlerquellen verzichtet, das saubere Programmieren ermöglicht und damit einen sehr gut lesbaren Code erstellt. Natürlich gibt es auch keine Regel, was nun als »sauber« zu bezeichnen ist. Python ist in gewisser Hinsicht aber sicherlich vorbildlich, da Anweisungsblöcke eingerückt werden *müssen*. Das Problem ist allerdings das Klassenkonzept. Ein weiteres Mal taucht ein Problem auf, wenn Klassen als etwas Anderes als Objekte betrachtet werden. Für diese seltsamen »Geistergebilde« gelten dann eigene syntaktische Regeln, und bei der Erstellung von Methoden gipfeln die Anforderungen in solchen Ungetümen wie

```
public static void main {
```

1.3 Ein mögliches Mißverständnis ...

Kritiken gegenüber anderen Sprachen wie C++ oder Java richten sich niemals gegen die Sprachen selbst, sondern vielmehr gegen eine gewisse Missionstätigkeit, die diesen Sprachen allein aus dem vermeintlichen Grund des »größeren Praxisbezugs« eine fahle Aura verleiht. »Die Praxis« gibt es genau so wenig wie »das Problem«. Kein Orakel, schon gar nicht das von »Delphi«, kann außerdem vorhersagen, wie die Situation in sechs Jahren aussehen wird. Es wäre schön, wenn dieses Tutorial diejenigen Kollegen kräftig unterstützen kann, die sich aus didaktischen Gründen der Java-Euphorie nicht anschließen und »neben der Spur« mit anderen Sprachen wie Python oder Ruby experimentieren. Eine Sprachendiskussion gehört auf die Ebene einer Arbeitsgemeinschaft und weniger in den Informatikunterricht. Wer Java oder C++ lernen will, tut es sowieso und wahrscheinlich besser als mit der Hilfe einer Schule.

2 Zeichenerklärungen

Dieses Symbol führt zur Mailadresse des Autors dieser Schrift (Heiko Schröder). Jede konstruktive Kritik, vor allem die Bereitschaft an der Mitarbeit an diesem Tutorial ist herzlich willkommen.


Durch Anklicken dieses Buttons erfolgt ein Download der entsprechenden PDF-Datei in dem Archivformat ZIP, das sich auf jedem Betriebssystem öffnen lassen sollte. Diese Archive enthalten gegebenenfalls auch ein Verzeichnis mit Beispieldateien.


Hier kann die gepackte Webseite heruntergeladen werden. Es handelt sich allerdings aus Platzgründen um ein TAR-Archiv, das mit dem Packer bzip2 gepackt ist. Derartige

Archive lassen sich nur unter Linux öffnen. Über die Konsole wird das Paket mit dem Befehl *tar xvfj <Paketname>* entpackt.

 Dieses Symbol führt zurück zu der Eingangsseite.

 Dieser Link führt zu www.squeakland.org, wo sich sehr viele Unterrichtsbeispiele und Modelle zu eToys finden.

 Dieser Link führt zur Originalseite www.squeak.org. Das Maskottchen weist auf die Disney-Productions hin, die das Morphic-Konzept in Squeak eingebaut hat.

3 Zu den Abteilungen des Tutorials

Das Tutorial enthält neun Teile, die im Folgenden kurz mit jeweils einem Screenshot beschrieben werden, falls sie schon existieren. Erst mit dem neunten Kapitel erfolgt die »offizielle« Freigabe. Die neun Kapitel enthalten den Kerninhalt. Erweiterungen in bezug auf »echte« Informatik sind für die Zeit nach der Freigabe geplant.

3.1 Reise in eine bunte Welt (Squeak 1)

Dieses Kapitel ist das einzig »theoretische«. Es werden die Grundlagen des objektorientierten Konzepts gelehrt. Im Zentrum stehen dabei die Begriffe Objekt (Instanz einer Klasse), Nachricht (message), Methode (method) und Klasse einschließlich ihres engen Naturbezugs. Die Abbildung 4 zeigt schematisch die Entstehung (Instanzierung) eines Objektes der Klasse Libellen durch ein anderes. In ergänzenden Abschnitten wird das »ältere« prozedurale Denken mit dem objektorientierten Denken verglichen.

Alter: Obere Mittelstufe (Klassen 9 und 10)

Bedeutung: Voraussetzung für alle Kapitel, die sich mit Smalltalk beschäftigen.

Umfang: 17 Seiten, 300 kB für PDF, 260 kB für Web

Webseite: Squeak 1: Eine Reise in eine bunte Welt

3.2 Squeak-Grundlagen (Squeak 2)

Das Kapitel vermittelt wichtige Handgriffe zur ersten Eingewöhnung. Es wird erklärt, wie Squeak installiert und gestartet wird, was die einzelnen Dateien bedeuten und vor allem wie die Maustasten belegt sind. Ferner wird das Speichern gezeigt. Die Squeak-Maus weist nach einem Klick mit der so genannten »blauen« Taste einen Halo, über den dieser so genannte Morph mit Hilfe bunter Handles verändert, kopiert, eingefärbt und dgl. werden kann. Außerdem wird der Begriff des Projektes besprochen.

Alter: keine Beschränkung

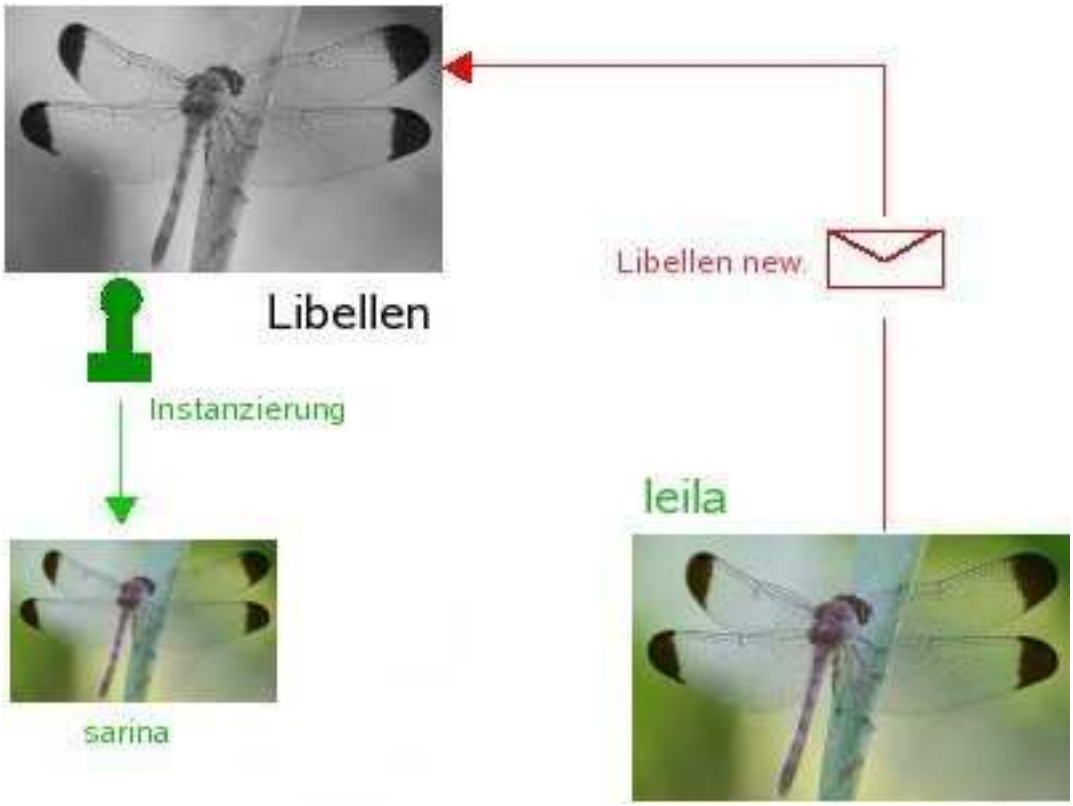


Abbildung 4: Instanzierung (Objektbildung) des Objekts sarina durch leila der Klasse Libellen

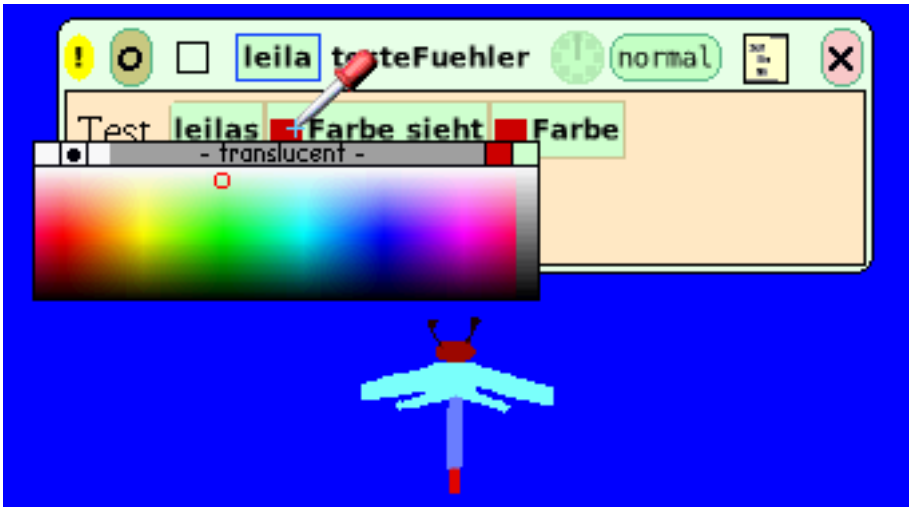
Abbildung 5: Der Halo um einen Morph (blaue Taste)

Bedeutung: Absolute Grundlage für alle praktischen Arbeiten.

Umfang: 10 Seiten, 320 kB für PDF, 300 kB für Web

Webseite: Squeak 2: Squeak-Grundlagen

3.3 eToys (Squeak 3)



EToys ist eine enorme Erweiterung der Turtle-Grafik von Seymour PAPERT. Die Arbeit mit eToys führt sozusagen behutsam in die objektorientierte Programmierung ein und ist auch Grundschulklassen zugänglich. Obwohl Smalltalk sich sehr eng an der Alltagssprache orientiert, sind bei eToys selbst die kleinsten syntaktischen Probleme beseitigt, indem Skripte durch das Ziehen von syntaktisch korrekten Kacheln zusammengestellt werden können. Die jungen (oder auch professionellen) Programmierer können sich allein auf das richtige Denken konzentrieren.

EToys ist allerdings nicht nur eine Erweiterung der Turtle-Grafik, sondern auch eine Erweiterung der alten Idee des Editors *Sketchpad* von Ivan SUTHERLAND aus den frühen sechziger Jahren: Bewegliche Grafiken können selbst gezeichnet werden und führen nach ihrer Fertigstellung gewissermaßen ein »Eigenleben«.

www.squeakland.org gibt ein eigenes Squeak-Image heraus, das erheblich abgespeckt, aber auf EToys optimiert ist. Dieses Image ist *nicht* mit der Vollversion zu verwechseln. Ebenso findet sich auf der Seite von Squeakland ein überreicher Fundus an EToys-Projekten für die Schule.

In Squeak 3 wird statt des allseits bekannten Autorennens ein anderes Projekt als »Höhepunkt« vorgestellt: Insekten am Seerosenteich. Wie immer in diesem Tutorial wird ein sehr enger Naturbezug angestrebt und das Projekt ist nur bis zu einer Stelle ausgeführt, von der aus ein eigentständiges Weiterarbeiten möglich ist.

Squeak 3 besteht aus den Teilen

1. Einleitung
2. Bearbeiten eines Morphs über Handles
3. Steuern eines Morphs über den Viewer

4. Steuern mehrerer Morphe: Insekten am Seerosenteich
5. Unterschiede zwischen eToys (Self) und Smalltalk*
6. Kleine Kontrollfragen

Die ersten drei Abschnitte führen immer tiefer in das Arbeiten mit eToys ein.

Der zweite Abschnitt konzentriert sich nur auf die Veränderungen des Morphs über den Halo und ist verhältnismäßig ausführlich. Der dritte Teil ist noch ausführlicher und beschreibt das Skripting mit *einem einzigen* Morph. Der vierte Abschnitt, der »Höhepunkt«, konzentriert sich bewußt auf das *Was* und überläßt dem Leser möglichst viele Freiheiten in bezug auf das *Wie*, sofern die Möglichkeiten der ersten drei Abschnitte nicht überschritten werden. Neues wird ausführlich erklärt. Zu diesem Teil des Tutorials gibt es mitgelieferte Projekte, die sich sowohl im PDF.zip als auch im TAR.BZ2 Archiv der Webseite befinden.

Alter: gesamte Sekundarstufe I; auch Grundschule (!), wenn der Text den Schülern nicht selbst überlassen wird.

Bedeutung: Erstklassige Einführung in das objektorientierte Denken. Sehr gut auch ohne Smalltalk-Kenntnissen möglich.

Umfang: 37 Seiten, ca. 1,2 MB PDF-Datei, ebenfalls 1,1 MB TAR-Archiv.

Squeak 3: eToys

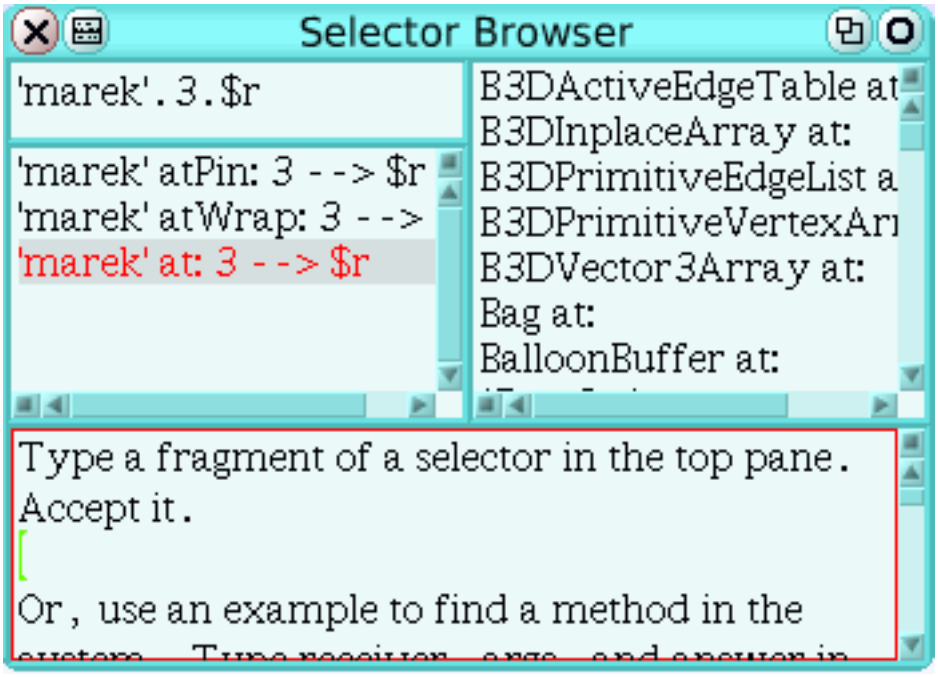


Abbildung 6: Der Methodenbrowser bei der Arbeit mit Smalltalk

3.4 Smalltalk-Hacking (Squeak 4)

Bei diesem Kapitel handelt es sich noch nicht um die Besprechung echter Informatik. Es wird die Struktur der Sprache besprochen und an einfachsten Beispielen direkt ausprobiert. Einige wesentliche Werkzeuge, wie der Methodenbrowser (siehe Abbildung) werden im Zusammenhang besprochen.

Alter: obere Mittelstufe

Bedeutung: Die Abschnitte ohne Stern sind Voraussetzung für Squeak 5.

Umfang: 41 Seiten, 440 kB PDF, 270 kB Web

Webseite: Squeak 4: Smalltalk-Hacking

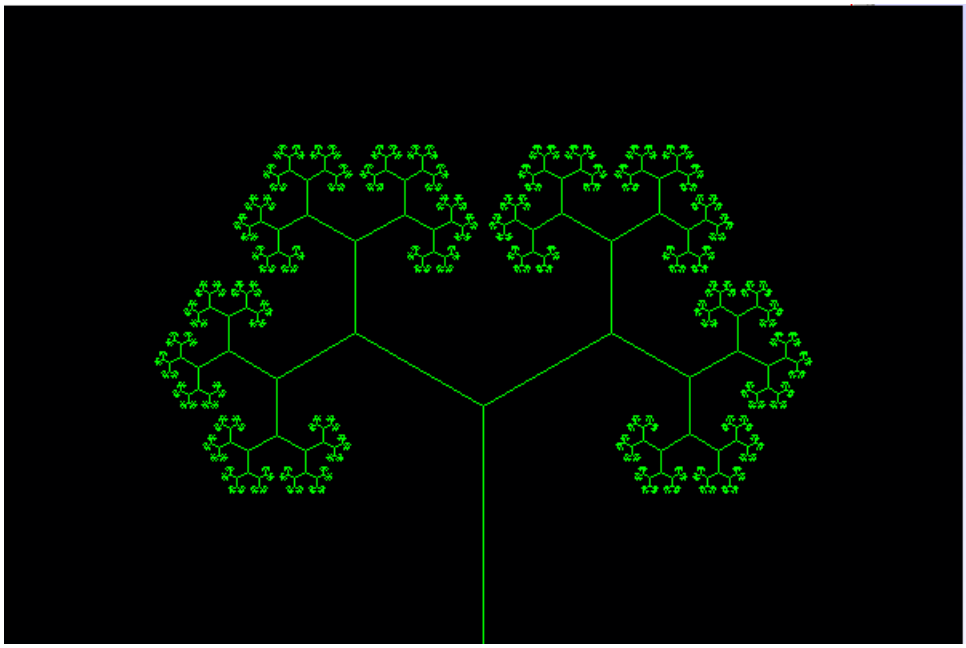
3.5 *Smalltalk I: Grundlagen (Squeak 5)*

echte Informatik mit Squeak (Objekt-Analyse, Objekt-Design, Objektorientierte Programmierung)

3.6 *Smalltalk II: Ergänzungen (Squeak 6)*

vor allem werden hier Kollektionen besprochen; der Charakter ist eine Minireferenz der wichtigsten Dinge

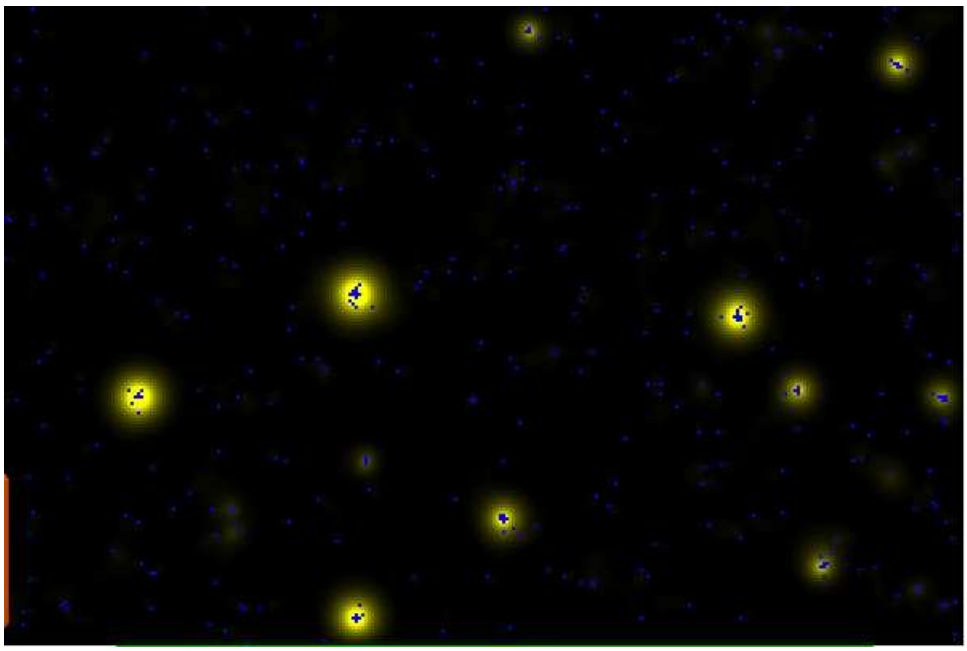
3.7 *Smalltalk III: Turtle-Grafik (Squeak 7)*



eToys läßt oft nur schwierig Rekursionen zu. Dieser Abschnitt des Tutorials stellt daher die herkömmliche Turtle in Form der Klasse *Pen* vor, wobei wir uns ausschließlich mit so genannten LINDEMAYER-Systemen befassen. Es geht weniger darum, *wie* die Turtle funkioniert, sondern eher darum, wie Rekursionen in Smalltalk realisiert werden können. Wieder ist – wie im ganzen Tutorial – der Naturbezug sehr stark. Lindemayer-Systeme sind Pflanzen, die nach einem bestimmten rekursiven Bauplan erzeugt werden können. Dabei wiederholt sich dasselbe Muster in beliebiger Schachtelungstiefe (Selbstähnlichkeit).

Alter: Rekursionen sind im allgemeinen kein besonders leichtes Thema. Die Einsatzmöglichkeit hängt entschieden davon ab, wie lange bereits Informatik unterrichtet wurde.

3.8 *StarSqueak oder Kedama (Squeak 8)*



Die nächste Arbeit, die begonnen wird, ist eine Beschreibung der Simulation von Selbstorganisation und Dezentralisierung. Eine Erweiterung der Turtle-Grafik von Logo führte Mitchell RESNICK zu StarLogo. Die Erweiterung besteht darin, Hunderte und Tausende von Turtlen gleichzeitig laufen zu lassen. Der Programmierer steuert aber keine einzelne Turtle selbst. Die Turtlen bewegen sich über einen Untergrund aus nicht sichtbaren Zellen, die – je nach Festlegung – irgendwelche Reize vermitteln, auf die die Turtlen ihr Verhalten verändern oder in spezieller Weise reagieren.

Diese Technik ermöglicht die Simulation von biologischen dezentralisierten Systemen. Dazu gehört die Entstehung von Schleimpilzen (siehe Abb.), die Ausbreitung einer Viruserkrankung, das Entstehen von Termitenhügeln oder die Ausbreitung eines Waldbrandes. Eine nicht biologische Simulation ist die Entstehung eines Verkehrsstaus. Diese Beispiele sind in StarLogo-Kreisen sehr gut bekannt und es ist verhältnismäßig leicht, darüber (allerdings auf Englisch) informiert zu werden. Wir versuchen eine Simulation zu erstellen, die die Entstehung der Formation eines Vogelschwarms ermöglicht. Auch dieses ist ein dezentralisiertes Problem: Es gibt in dem Vogelschwarm keinen »Leitvogel«, der alles unter Kontrolle hat. Dabei benutzen wir StarLogo in der eToys-Form Kedama.

Von Marcus Denker erfuhr ich, dass StarSqueak wegen des leistungsfähigeren Kedamas nicht weiterentwickelt wird, weshalb dieser Teil des Tutorials sehr wahrscheinlich auf Kedama aufbauen wird. Es gibt allerdings schon eine sehr gute deutsche Anleitung, die bei www.squeak.de unter *Dokumentation* ▷ *Tutorials* zu finden ist (deutsche Übersetzung des englischen Originals). Wir sind damit blendend zurechtgekommen. Wir konzentrieren uns daher auf das – vielleicht weniger bekannte – Beispiel des Vogelzugs.

Alter: Auf jeden Fall obere Sekundarstufe I und Oberstufe

noch keine Snapshots

3.9 *Alice 3D oder entsprechender Ersatz (Squeak 9)*

ein großer Spaß im Dreidimensionalen