Einführung in Squeak

Rita Freudenberg

Installation Squeak ist eine offene, kostenlose und multimediale Entwicklungsumgebung, die auf der Programmiersprache "Smalltalk" aufsetzt und maßgeblich von Alan Kay mitentwickelt wurde. Es gibt verschiedene Umgebungen in Squeak, in diesem Kurs werden wir die Entwicklungsumgebung "Etoys" verwenden, die für Kinder und Programmieranfänger gedacht ist.

Mit "Etoys" ist es möglich, innerhalb sehr kurzer Zeit ohne Vorkenntnisse ein Programm zu entwickeln.

Squeak ist für über 20 Plattformen verfügbar, u.a. für Windows (ab 95/98), MacOS 7.5-9.2, MacOS X und Unix/Linux. Für eine Installation werden jeweils 4 Dateien benötigt: die virtuelle Maschine (plattformabhängig), das Image (hier sind alle Objekte gespeichert), die Sources (Quelltexte des letzten großen Release) und die Changes (Änderungen der Quelltexte gegenüber der Sources). Auf der CD befinden sich die Installationsdateien für Windows, Linux und MacOS, für weitere Betriebsysteme kann man unter "http://www.squeakland.org" unter dem Menüpunkt"Download Squeak" die Installationsdateien herunterladen. Als einführende Beispiele sind 3 Tutorials verfügbar, außerdem ein Hand-Out, welche als .pdf auch auf der CD enthalten sind.

Einführung Nach dem Starten von Squeak öffnet sich ein Fenster mit einer orangen und einer roten Klappe am unteren Bildschirmrand. Durch Anklicken kann man diese Klappen öffnen. In der Navigatorklappe sind die Navigationsmenüs und das Malwerkzeug zu finden, außerdem kann man die Sprache und die Lautstärke einstellen. In der Lagerklappe befinden sich nützliche Objekte, die man später für seine Projekte benutzen kann. Sollten nicht alle Objekte in der Lagerklappe sichtbar sein, kann man einfach am Griff der Klappe anfassen und diese weiter heraus ziehen. Ein weiterer Klick auf die Klappe schließt sie wieder.

In der Etoys-Umgebung von Squeak programmiert man, indem man das Verhalten beschreibt eines Objektes. In der Regel erhält man ein Objekt, indem man es sich malt, mann kann aber auch Bilder aus anderen Dateien laden. Um ein Bild zu malen, klickt man auf das Icon für das Malwerkzeug in der Navigator-Klappe (ein Pinsel). Der Umgang mit dem Malwerkzeug wird ausführlich im ersten Tutorial erklärt.

Um dem Objekt nun ein Verhalten zu geben öffnen wir seinen Betrachter. Dafür gibt es die "Smarties". Sie erscheinen, wenn man die Maus auf das Objekt bewegt und einen Moment wartet oder wenn man Alt-Linksklick (unter Windows) auf das Objekt ausführt.

Die Bedeutung der einzelnen Smarties wird im Tutorial 2 erläutert. Um einen Betrachter zu öffnen, klickt man auf das türkise Smartie mit dem Auge. Nun sieht man am rechten Bildschirmrand einen hellgrünen Bereich mit Informationen über das Objekt und vielen Kacheln. Erste Schritte für die Arbeit mit dem Betrachter und das Erstellen von Skripten sind im Tutorial 3 beschrieben.

Nach dem Durcharbeiten der drei Tutorials besitzt man das Handwerkszeug, um konkrete Squeak-Projekte zu erstellen.

Ein Projekt in Squeak Wenn man ein Projekt in Squeak entwickelt geht es nicht in erster Linie darum, die Benutzung von Squeak an einem Beispiel zu lernen sondern darum, mehr über das Beispielthema zu lernen, indem man Squeak benutzt. Ausgesuchte Phänomene des Beispielthemas werden in Squeak nachgebildet. Das jeweilige Thema kann aus einem beliebigen Anwendungsfach kommen, beispielsweise Physik. Dort gibt es für das Profilfach Physik das Thema "Gravitationsfeld", was auch als Thema im Wahlpflichtfach Physik vorkommt. Es eignet sich gut, bestimmte Funktionen von Squeak auszuprobieren.

Das Ziel des Workshops ist es nicht, ein komplett fertiges Projekt zu erstellen (wobei wir natürlich soweit wie möglich kommen wollen). Ich möchte Ihnen am Beispiel die Möglichkeiten für den Einsatz von Squeak nahebringen und hoffentlich soviel Interesse wecken, dass Sie zuhause und in der Schule damit weitermachen.

Systembeschreibung Um in Squeak den Fall eines Balles unter Berücksichtigung der Schwerkraft abzubilden, benötigen wir zuerst eine graphische Repräsentation des Balles und eine Möglichkeit, sein Verhalten zu beschreiben. Dazu ist es sinnvoll, sich zunächst zu überlegen, was das Wesentliche seines Verhaltens ist. Die Modellierung erfolgt über das Beschreiben dieses Verhaltens. Dazu sind keine Vorkenntnisse nötig, jeder Schüler hat schon einmal mit einem Ball gespielt und sollte beschreiben können, was im Einzelnen passiert. Man könnte es zum Beispiel so formulieren:

Wird ein Ball aus einem beliebigen Abstand über dem Boden losgelassen, fällt er nach unten, und zwar mit einer gewissen Beschleunigung. Sobald er den Boden berührt, prallt er ab und bewegt sich wieder nach oben, diesmal wird er während der Bewegung nicht schneller sondern langsamer. Irgendwann ist die Bewegung gleich Null und er fällt wieder nach unten. So geht es eine ganze Weile, wobei er bei jedem Aufprall auf dem Boden eine kürzere Strecke nach oben springt, bis er schließlich auf dem Boden liegen bleibt.

Umsetzung in Squeak Die graphische Repräsentation malt man einfach mit dem Malwerkzeug. Es ist in Squeak auch möglich Bilder zu importieren, die dann genauso verwendet werden können wir selbstgemalte Objekte. Für den gemalten oder importierten Ball holt man sich nun die Smarties und mit Klick auf das türkise Smartie den Betrachter. Nun befindet man sich in der Skripting-Umgebung, in der man das Verhalten beschreiben kann.

Der erste Teil unserer Beschreibung sagt, dass der Ball beschleunigt nach unten fällt. Wir ziehen uns entweder eine Kachel "leeres Skript" auf die Oberfläche oder beginnen

direkt mit der ersten Anweisungskachel, die auch ein neues Skript erzeugt, wenn man sie auf die Oberfläche legt. Es ist empfehlenswert, dem Skript gleich einen aussgekräftigen Namen zu geben, z.B. “falle”. Um den Ball nach unten fallen zu lassen, holen wir uns aus der Bewegungskategorie eine Kachel “gehe vorwärts um” und packen sie in das Skript. Dort besteht die Möglichkeit, einen bestimmten Wert anzugeben.

Die Abarbeitung eines Skriptes erfolgt, indem man das Skript anschaltet (startet). Dann wird es wieder und wieder durchlaufen, bis es wieder abgeschaltet (gestoppt) wird. Man kann sich vorstellen, dass das Objekt, dem das Skript zugeordnet ist, “zum Leben erweckt” wird. Es ist auch möglich, ein Skript genau einmal ablaufen zu lassen, das wird erreicht, indem man auf das gelbe Feld mit dem Ausrufezeichen im Skript klickt.

Mit diesem Wissen im Hinterkopf wird schnell klar, was passiert, wenn man in der “gehe vorwärts um”- Kachel einen Wert eingibt: der Ball bewegt sich um eine bestimmte Strecke vorwärts, und zwar bei jedem Durchlauf des Skriptes um die gleiche Strecke. Jeder Durchlauf des Skriptes startet auch nach einer vorgegebenen, immer gleichen Zeit (die man ändern kann, indem man im Skript auf die Uhr klickt und die Maustaste einen Moment gedrückt hält. Es öffnet sich ein Menü, aus dem man auswählen kann, wie oft pro Minute das Skript ablaufen soll.). Das heisst, der Ball bewegt sich ständig vorwärts, aber in einer gleichförmigen, nicht in einer beschleunigten Bewegung. Das lässt sich einfach nachprüfen, indem man das Skript, so wie es jetzt ist, startet und den Ball beobachtet. Vermutlich bewegt sich der Ball jetzt noch die falsche Richtung. Dazu muss man wissen, dass “gehe vorwärts um” immer relativ gemeint ist. Vorwärts bedeutet in jedem Fall die Richtung, in die das Objekt “schaut”. Wenn man die Smarties für ein Objekt holt, sieht man in der Mitte des Objektes einen grünen Pfeil, der in eine Richtung zeigt. Das ist vorwärts. Man kann die Richtung ändern, indem man mit der Maus den Pfeil dreht. Der Wert für “Richtung” des Objektes (in der Einfach und der Bewegungskategorie) hat den Wert “0” bei vorwärts. Man kann die Blickrichtung auch ändern, indem man das Objekt dreht (mit der “drehe dich um”-Kachel). Dann heisst “vorwärts” auch, in die neue Blickrichtung zu gehen, aber der Wert für “Richtung” hat sich geändert. Wenn man ein bisschen mit der “drehe dich um”-Kachel rumspielt, wird man schnell merken, dass die Einheit, um die man sich dreht, immer in Grad angegeben wird (bei “drehe dich um 360 landet man also wieder in der Ausgangsstellung).

Um sich in die der Blickrichtung entgegengesetzte Richtung zu bewegen, kann man einfach negative Werte in der “gehe vorwärts um”-Kachel verwenden. Man kann den Ball also nach unten fallen lassen, indem man negative Werte benutzt, die Richtung mit dem grünen Pfeil ändert oder das Objekt mit einer Kachel dreht. Da wir wissen, das sich die Bewegungsrichtung in unserem Beispiel noch mehrmals ändern wird, aber nur immer in entgegengesetzte Richtungen, ist die Verwendung positiver oder negativer Werte die flexibelste Lösung.

Achtung: Falls dabei der Ball den sichtbaren Bereich des Bildschirmes verlassen sollte, gibt es eine “Notfallprozedur” um ihn wieder “einzufangen”. Man klickt “Escape”, wodurch sich das Weltmenü öffnet. Dort wählt man den Eintrag ”Spielwiese einrichten” und dann “Irrläufer fangen”. Nun sollte das verschwundene Objekt links oben auf dem Bildschirm wieder sichtbar sein. Falls man es aus Versehen gelöscht hat, kann man einfach in der Mülltonne nachsehen, indem man sie anklickt, und bei Bedarf das Objekt

wieder auf den Bildschirm schiebt.

Wie kann man diese Bewegung beschleunigen? Bei jedem Skriptaufruf müsste eine größere Strecke zurückgelegt werden als beim vorherigen Aufruf. Das ist erreichbar, wenn man anstelle eines festen Wertes eine Variable benutzt, der man dann in jedem Schritt einen neuen Wert zuweisen kann.

Eine Variable erzeugt man, indem man im Betrachter auf das Feld mit dem "v" vor dem Namen des Objektes klickt. Man kann einen Namen für die Variable angeben, die dann in einer neuen Kategorie "Variablen" im Betrachter erscheint. In unserem Fall nennen wir die Variable "Geschwindigkeit". Vor der Kachel der Variablen ist ein kleines Icon, wenn man darauf klickt, erhält man ein Menü, das es erlaubt, den Datentyp der Variablen zu ändern. Das ist jetzt aber nicht nötig, da standardmäßig ein numerischer Datentyp eingestellt ist, den wir hier auch haben wollen.

Um den Weg zu berechnen, der nach einer bestimmten Zeit zurückgelegt wurde, brauchen wir den Weg zum Zeitpunkt t und die Geschwindigkeit, mit der sich das Objekt bewegt. Addieren wir diese Geschwindigkeit zum Weg dazu, erhalten wir den Weg zum Zeitpunkt t+1. Das setzen wir in Squeak um, indem wir eine Kachel "Kugels yäus der Bewegungs-Kategorie holen (am grünen Pfeil anfassen - Zuweisungskachel). Da unser Ball von oben nach unten fallen soll, repräsentiert das y unseren Weg. Zu diesem Weg addieren wir die Geschwindigkeit hinzu indem wir hinter die zuerst eine Kachel mit "Kugels y"(diesmal vorne anfassen, wir brauchen den Wert) in die Zuweisungskachel einfügen, dann auf den kleinen grünen Pfeil am Ende klicken, so dass sich ein Feld öffnet, indem wir ein "+ëinstellen, und dann eine Kachel mit dem Wert der Geschwindigkeit anfügen. So haben wir eine Zeile "Kugels y ergibt sich aus Kugels y + Kugels Geschwindigkeit". Nun stellt sich noch die Frage nach der Geschwindigkeit. Physikalisch ist die Geschwindigkeit die erste Ableitung der Beschleunigung, d.h., die Änderung der Geschwindigkeit von einem Schritt zum nächsten ist die Beschleunigung.

In unserem Beispiel erstellen wir uns eine zweite Variable namens "Beschleunigung". Dieser weisen wir einen negativen Wert zu, da wir ja wollen, dass der Ball nach unten fällt, was in Squeak bedeutet, dass der Wert für y sich verringern muss. Wir fügen unserem Skript eine weitere Zeile hinzu"Kugels geschwindigkeit erhöhen um Kugels Beschleunigung". Das führt dazu, dass in jedem Schritt die dann aktuelle Geschwindigkeit um den Wert der Beschleunigung erhöht wird, also die Geschwindigkeit immer größer wird.

Mit einer kleinen Ergänzung haben wir eine fallende Kugel definiert: wir müssen die Bewegung anhalten, sobald die Kugel den Boden berührt. Dazu brauchen wir einen Test-Block. Den erhält man, wenn man auf das rechteckige gelbe Icon im Kopf des Skriptes klickt. Den Testblock fügen wir an das Ende unseres Skriptes an. Wie an der Struktur zu erkennen ist können wir hier erfragen, ob eine Bedingung erfüllt ist oder nicht und in Abhängigkeit vom Ergebnis bestimmte Aktionen ausführen (also eine Alternative bzw. IF-Anweisung).

Wenn wir testen wollen, ob der Ball den Boden berührt, brauchen wir erstmal einen Boden. Eine gute Möglichkeit ist es, eine "Spielwiese" zu verwenden, wie sie im Lager zu finden ist. Das ist im Prinzip ein farbiges Rechteck mit einigen zusätzlichen Eigenschaften gegenüber einfachen, selbstgemalten Objekten. Die Spielwiese holt man aus dem Lager,

indem man dieses weit genug öffnet und die Spielwiese mit der Maus herauszieht. Legt man nun den gemalten Ball mit der Maus in die Spielwiese, gehört dieser nun zur Wiese, die Spielwiese ist sein “Eigner”(Besitzer).

Am einfachsten ist es, wenn wir testen, ob y einen bestimmten Wert unterschreitet. Wir holen uns eine Kachel mit dem Wert von “Kugels yünd fügen sie hinter Testïn den Testblock ein. Mit den kleinen hoch-und-runter-Pfeilen wählen wir den gewünschten Vergleichsoperator und den Wert zum Vergleich aus. In den “Ja-Zweig des Testblockes schreiben wir nun, was passieren soll, wenn die Bedingung erfüllt ist: der Geschwindigkeit wird der Wert ”0ßugewiesen. Um sicherzustellen, dass dann nicht im nächsten Durchlauf des Skriptes die Geschwindigkeit wieder erhöht wird und so der Ball weiterfällt, wird die Befehlszeile, die die Geschwindigkeit berechnet, in den NeinZweig des Testblockes verschoben.Um die Bewegung des Balles und besonders die Beschleunigung darzustellen kann man ein Bewegungsdiagramm zeichnen. Wie man das macht, ist im Anhang im entsprechenden Projekt zu finden.

An dieser Stelle gibt es vielleicht schon einige, die es nervig finden, jedesmal den Ball wieder an den Ausgangspunkt zurüclzulegen und die Anfangsgeschwindigkeit einzustellen. Das einfachste ist es, hierfür ein Reset-Skript zu schreiben. Dazu holt man aus der Skripte-Kategorie eine Kachel leeres Skriptünd benennt sie um in reset“. Jetzt legen wir Kacheln für alle Anweisungen, die wir bisher per Hand gemacht haben in das Skript. Das wäre also eine Zuweisungskachel für die Geschwindigkeit und je eine Kachel für “Kugels xünd “Kugels yäus der Kategorie Bewegung. Die Werte für x und y kann man am einfachsten erhalten, indem man den Ball an die gwünschte Stelle legt und dann die Kacheln nimmt. Ist das Skript fertig, klickt man auf den blau umrandeten Eigentümer des Skriptes im Skriptkopf und erhält ein Menü. Dort wählt man den Eintrag “Knopf um dieses Skript auszuführenünd erhält einen Knopf mit dem Namen des Skriptes. Wenn man diesen anklickt, wird das Skript einmal ausgeführt. Einen solchen Knopf kann man für jedes Skript über dieses Menü holen, wenn man einen braucht.

Aber wir wollten ja einen Ball fallen lassen. Wenn dieser den Boden berührt, ändert er seine Richtung und bewegt sich wieder noch oben, er bleibt nicht liegen. Also ändern wir unser Skript. Wir führen neben der Geschwindigkeit noch einen Faktor ein. Dieser Faktor ist kleiner als 1, und er realisiert den Verlust an Geschwindigkeit, der auftritt, wenn der Ball den Boden berührt. Der Ball bewegt sich ja nicht mit der gleichen Geschwindigkeit nach oben, mit der er nach unten gefallen ist. Also multiplizieren wir die Geschwindigkeit mit einem Faktor kleiner als 1, um diesen Verlust darzustellen. Außerdem geben wir dem Faktor ein negatives Vorzeichen. Dadurch wird die Richtung, in die die Geschwindigkeit wirkt, umgedreht, d.h., der Ball bewegt sich danach in die entgegengesetzte Richtung. Was genau ändern wir also? Zuerst können wir die Befehlszeile aus dem NeinZweig nach oben über den Testblock verschieben, wir wollen ja nicht mehr anhalten, wenn wir unten sind. Dann müssen wir natürlich auch ändern, was passieren soll, wenn der y-Wert erreicht oder unterschritten wird, der anzeigt, dass der Ball den Boden berührt. In diesem Fall berechnen wir die Geschwindigkeit neu aus der alten Geschwindigkeit multipliziert mit dem Faktor. Wenn wir jetzt das Skript starten sehen wir schon den Ball springen, allerdings gibt es noch einen kleinen SSchönheitsfehler“: Angenommen, wir haben als y-Wert für den Boden die ”1ängegeben. Irgendwann wird die Geschwindigkeit sehr klein

und unterschreitet diesen Wert. Das bedeutet, nach dem Aufprall bewegt sich der Ball um weniger als "1nach oben. Dann folgt der Test, der angibt, dass der y-Wert von 1 unterschritten wurde, und die Bewegung nach oben gleicht nicht mehr die Bewegung nach unten aus. Der Ball würde langsam immer weiter nach unten fallen.

Deshalb fügen wir in den "JaZweig noch einen Testblock ein, der abfragt, ob die Geschwindigkeit kleiner als "1ïst und dann das Skript anhält. Zu dieser Simulation kann man weitere Bewegungsdiagramme zeichnen, beispielsweise ein Weg-Zeit-Diagramm und ein Beschleunigungs-Zeit-Diagramm und so die durch die pragmatische Herangehensweise gewonnen Ergebnisse auf die bekannten Darstellungsformen übertragen, aber mit einem tieferen Verständnis für die Zusammenhänge.

Literatur Mark Guzdial "Squeak - Object-Oriented Design with Multimedia Applications", Prentice Hall 2001