CaveManMX – Computerspielkonzepte in der Multimedia-Programmierausbildung

Malte Beyer, Christian Geiger
Hochschule Harz, Medieninformatik
cgeiger@hs-harz.de
www.medieninformatik.de

Abstract: Zur Unterstützung der Programmierausbildung im Bereich Medieninformatik wird ein Spielkonzept vorgestellt, das den Teilnehmern von Veranstaltungen im Bereich Medienprogrammierung zusätzliche Motivation zur Beschäftigung mit dieser Materie bietet. Es wurde ein System entwickelt, bei dem die Teilnehmer ihren eigenen Spielcharakter multimedial erstellen und im Rahmen eines Wettkampfes gegeneinander antreten lassen. Die Umsetzung erfolgt mit dem im Medienbereich als Quasi-Standard etablierten Macromedia Flash, das mit der Version 2.0 der Scriptsprache ActionScript erstmals die von etablierten Programmiersprachen (z. B. Java) bekannten objektorientierten Konzepte anbietet.

1 Motivation

Das Projekt „CaveManMX“ wurde an der Hochschule Harz mit der Absicht initiiert, Studierenden der Medieninformatik und verwandter Disziplinen den Einstieg in die Multimedia-Programmierung zu erleichtern. In der Vergangenheit hat sich gezeigt, dass die bisherigen Ansätze der Programmierausbildung zu einer geringen Motivation unter den Studierenden der Medieninformatik führten. Ein Grund dafür ist sicherlich das Ungleichgewicht zwischen dem Programmieraufwand und dem daraus resultierenden visuellen Ergebnis. Es fehlt zu Beginn der Ausbildung ein Motivationsfaktor, welcher den vermeintlich hohen Aufwand der Programmierung mit einem ansprechenden Ergebnis belohnt. „CaveManMX“ bietet den Studierenden auf spielerische Art und Weise ein ansprechendes visuelles Feedback auf ihre ersten Programmierschritte. Die Teilnehmer sollen dabei ein Gespür für den richtigen Umgang mit verschiedenen programmierspezifischen Charakteristika strukturierter und objektorientierter Programmierung entwickeln. Zusätzlich sollen auch elementare Konzepte multimedialer Programmierung berücksichtigt werden, z. B. Integration unterschiedlicher Medien und grafische Interaktion. Ziel ist es, einen eigenen Spielercharakter zu programmieren und mit möglichst „intelligentem“ Verhalten auszustatten, indem der Studierende eine ihm zur Verfügung gestellte Befehlsbibliothek nutzt. Darüber hinaus steht der volle Funktionsumfang der benutzten Programmiersprache ActionScript 2.0 zur Verfügung. Der fertige Charakter wird später in die Applikation eingebunden und tritt in einem Wettkampf gegen Charaktere anderer Teilnehmer an. Erfahrungsgemäß haben derartige spielbasierte Aufgaben in anderen Veranstaltungen stets ein deutlich höheres Aktivierungspotential bei den Teilnehmern als klassische Aufgabentypen. Obwohl als Zielgruppe klar Studierende der ersten Semester angesprochen werden sollen, steht einem Einsatz in Veranstaltungen des Hauptstudiums nichts entgegen, da auch komplexe KI-Strategien und kooperatives Verhalten unterstützt werden.

2 Verwandte Arbeiten

Die Idee, spielerische und/oder visuelle Konzepte zur Programmierung bzw. Programmierausbildung zu nutzen, ist nicht neu. Das an der ETH Zürich entwickelte Programm „Kara“ versucht beispielsweise dem Nutzer durch einen programmierbaren Marienkäfer einen spielerischen Zugang zu den Grundlagen der Informatik zu vermitteln [HNR03]. Dabei soll der Nutzer spielerisch ein grundlegendes Verständnis der Prinzipien erwerben, auf denen moderne Programmierung basiert. Seit den 90er Jahren gibt es verschiedene Ansätze visueller Programmierung, teilweise direkt zugeschnitten auf Kinder. Ken Kahn’s „ToonTalk“ ist dabei einer der interessantesten Ansätze visueller Programmierung [Ka96]. Basierend auf den Ideen der Constraint Logischen Programmierung entwickelte Kahn eine vollständig visuelle Metapher logischer Programmierung, die eine Programmentwicklung durch Beispiele („Programming by Example“) ermöglicht. Einen ähnlichen Ansatz für Produktionsregeln verfolgte das Cocoa/KidSim-Projekt von Apple [CS95]. Programme entstehen durch visuelles „Vormachen“, d.h. durch Angabe einer visuellen Vorbedingung („Affe steht vor Kiste“) und Nachbedingung („Affe springt auf Kiste“). Dadurch ließen sich einfache Animationen und Spiele von Kindern (ab ca. 8 Jahre) entwickeln. Die in CaveManMX angewandte Wettbewerbsidee findet man schon im „Core Wars“ Projekt [CW], bei dem konkurrierende Programme, geschrieben in einer Art reduziertem Assemblercode (redcode) in einem simulierten Speicher versuchen, sich gegenseitig zu löschen. Als direkte Anregung für das hier beschriebene Projekt diente IBM’s RoboCode [RobC]. Hier versuchen virtuelle Panzer mit in Java entwickelten Strategien, sich gegenseitig zu bekämpfen. Im Unterschied zu diesem Ansatz konzentriert sich CaveManMX auf eine nicht-kriegerische Spielidee, die auch kooperatives Handeln zwischen den Charakteren, die Integration multimedialer Inhalte, sowie (in einem weiteren Projekt) die Nutzung einer Client-Server Architektur für Netzwerkgaming mittels Flash Communication Server ermöglicht. Als Basis wird nicht Java sondern ActionScript 2.0 / Flash MX 2004 verwendet.

3 Spielidee

Bei CaveManMX besitzt jeder Höhlenmensch eine eigene vom Studierenden programmierte Strategie. Zu Beginn jeder Runde starten alle Höhlenmenschen an ihrem zufällig positionierten Lagerfeuer. Ziel jeder Runde ist es, als erster drei unterschiedliche Nahrungsmittel (Fleisch, Beeren, Pilze) einzusammeln und zum eigenen Lagerfeuer zu bringen. Jeder Höhlenmensch kann dazu vorwärts und rückwärts laufen, sich drehen sowie sich in einem eingeschränkten Bereich umschauen, um nach Lebensmitteln, Lagerfeuern, Dinosauriern oder anderen Höhlenmenschen Ausschau zu halten. Andere Höhlenmenschen können den Spieler überfallen und ausrauben, indem sie ihm sein transportiertes Nahrungsmittel entwenden. Ein kooperatives Handeln ist jedoch auch möglich. Zusätzlich bewegt sich ein Dinosaurier über das Spielfeld und macht Jagd auf ihm zu nahe kommenden Höhlenmenschen. Derjenige Höhlenmensch, der die meisten Punkte durch schnelles Sammeln und wenige eigene Treffer erzielen konnte, gewinnt das Spiel.

4 Befehle und Ereignisse

Um seinen Höhlenmensch geschickt über das Spielfeld zu bewegen, stellt CaveManMX dem Studierenden neben der eigentlichen Programmiersprache ActionScript 2.0 unterschiedliche spezielle Befehle in einer API zur Verfügung. So kann der Höhlenmensch frei über das Spielfeld bewegt werden, kann sich unter anderem drehen, sich umschauen, Steine werfen und Lebensmittel aufnehmen. Über die entsprechenden Befehle kann der Studierende außerdem spielinterne Informationen erlangen, wie z.B. Informationen über die Anzahl der teilnehmenden Höhlenmenschen, die Anzahl der Runden oder die Zahl der aktuellen Runde. Diese Befehle nutzen dem Studierenden jedoch erst, wenn er mit diesen auf entsprechende Ereignisse reagieren kann. Diese Ereignisse ergeben sich aus den Handlungen des Höhlenmenschen, wie z.B. der Kollision mit dem Spielfeldrand oder dem Entdecken von Nahrungsmitteln auf dem Spielfeld. CaveManMX berechnet hierfür zeitgenau die entsprechenden Ereignisse und sendet diese an den betroffenen Höhlenmenschen. Dieser kann dann anhand der übermittelten Ereignisse wichtige Informationen darüber erlangen. So erhält ein Höhlenmensch beispielsweise beim Erblicken eines Nahrungsmittels ein *FoodInSightEvent*, welches Informationen über die Position, die Peilung, die Distanz sowie den Typ des Nahrungsmittels enthält. Durch dieses Ereignismodell wird jeder Höhlenmensch zum passenden Zeitpunkt mit den nötigen Informationen versorgt. Es obliegt dem Studierenden, diese Informationen in seiner Programmierung richtig auszuwerten und seinen Höhlenmenschen entsprechend darauf reagieren zu lassen. Weitere wichtige Ereignisse sind z. B. das Erkennen anderer Spielobjekte wie Lagerfeuer, Dinosaurier oder andere Höhlenmenschen sowie Treffer von anderen Mitspielern.

5 Technische Umsetzung

CaveManMX wurde komplett mittels Flash MX 2004 Professional von Macromedia umgesetzt, welches als eines der wichtigsten Werkzeuge in der Multimedia Programmierung gilt. Mit der Veröffentlichung von Flash MX 2004 wurde die bisherige Programmiersprache ActionScript 1.0 durch ActionScript 2.0 abgelöst, was den Funktionsumfang von Flash um zentrale objektorientierte Konzepte erweitert. So unterstützt ActionScript seit der Version 2.0 das Konzept der objektorientierten Programmierung, welches die Verwendung von Klassen, Vererbung, Interfaces, Packages und weiteres erlaubt. Bei CaveManMX entwickelt der Studierende seine Höhlenmenschen textbasiert mittels ActionScript. Voraussetzung hierfür ist die Verwendung von Macromedia Flash MX 2004 oder Professional, welches jedoch in der praktisch orientierten Medieninformatik-Ausbildung stets vorhanden sein sollte. Anhand eines Template können die Studierenden in einer eigenen externen ActionScript-Datei die Implementierung ihres Höhlenmenschen vornehmen. Dabei stehen die von CaveManMX angebotene API sowie der volle Funktionsumfang von ActionScript 2.0 inklusive des Debuggers zur Verfügung. Nach Beendigung der Programmierung und dem Export als .swf Datei, kann diese in den Simulator geladen werden, in dem der Wettkampf beginnt. Das CaveManMX-Template stellt den Studierenden sowohl eine Methode zur Verfügung, die zu Beginn jeder Runde ausgeführt wird (*startUpAction*), als auch eine Methode, welche bei fehlenden Events in

einer Schleife abgearbeitet wird (*doThis*). Methoden, die bei auftretenden Events angesprochen werden, können optional in den Code aufgenommen und nach eigenen Wünschen angepasst werden. Im Folgenden zeigt ein einfaches Beispiel einen (nicht sehr intelligenten) Höhlenmenschen, der sich zu Beginn jeder Runde zum oberen Spielfeldrand ausrichtet und stets ein Rechteck von 100 Pixeln im Uhrzeigersinn abläuft.

```
import caveManClasses.CaveMan;
class MyCaveMan extends CaveMan{
	function MyCaveMan (timeline:MovieClip, depth:Number, caveManName:String, arrayID:Number, ui:Object) {
		super(timeline, depth, caveManName, arrayID, ui);
	}
	// wird einmalig zu beginn jeder runde ausgeführt
	public function startUpAction():Object {
		var cmd:Object = {
			step_1:function(me:Object) { me.turnLeft(me.getHeading()); }
		}
		return cmd;
	}
	// standardfunktion, welche bei nicht vorhandenen events in einer schleife ausgeführt wird
	public function doThis ():Object {
		var cmd:Object = {
			step_1:function(me:Object) { me.goAhead(100); me.turnRight(90); }
		}
		return cmd;
	}
}
```

Um die „Intelligenz“ des Höhlenmenschen zu erhöhen, könnte man den Charakter Ausschau nach anderen Objekten halten lassen. Zu diesem Zweck könnte er beispielsweise seine *doThis()* Funktion mit den Zeilen *me.turnHeadRight(60), me.turnHeadLeft(60)* ergänzen. Dies veranlasst den Höhlenmenschen, sich um 60° nach rechts umzuschauen. Durch das Umschauen nimmt der Höhlenmensch Gegenstände auf dem Spielfeld wahr, über die ihn CaveManMX mittels Events informiert. Entdeckt der Höhlenmensch beim Umschauen beispielsweise ein Nahrungsmittel, so erhält der Höhlenmensch vom System ein *FoodInSightEvent*, auf das der Höhlenmensch wie im folgenden Beispiel reagieren kann.

```
public function onFoodInSightEvent(e:Object):Object {
	var cmd:Object = {
		step_1:function(me:Object) {
			me.out("nahrungsmittel vom typ " + e.getType() + " auf " + e.getX() + "|" + e.getY() + " gefunden!");
		},
		step_2:function(me:Object) {
			e.getBearing() < 0 ? me.turnLeft(-e.getBearing()) : me.turnRight(e.getBearing());
			me.goAhead(e.getDistance());
			me.takeFood();
		}
	}
	return cmd;
}
```

Selbst kooperative Strategien von mehreren Höhlenmenschen sind auf diese Art möglich. Ein Höhlenmensch, der sich auf das schnelle Finden und Sammeln von Nahrungsmitteln spezialisiert hat, könnte sich beispielsweise mit einem zweiten Höhlenmenschen verbünden, welcher eher auf das Werfen von Steinen optimiert wurde. Dieser würde ihn (gegen Belohnung) vor etwaigen Angreifern schützen. Auch die Entwicklung lernender Höhlenmenschen ist bei CaveManMX möglich. Während eines Spiels, welches aus mehreren Runden besteht, könnte sich ein Höhlenmensch merken, welcher Höhlenmensch ihm bislang den größten Schaden zugefügt hat und diesen in den kommenden Runden meiden.

6 Resultat und Ausblick

Zur Überprüfung der eigenen Programmierung stehen in CaveManMX einige Tools zur Verfügung. Während das Spiel sowohl aus einer zweidimensionalen Vogelperspektive, als auch aus einer isometrischen dreidimensionalen Perspektive beobachtet werden kann, geben verschiedene optionale Menüs Auskunft über wichtige Attribute des Höhlenmenschen, wie z. B. Position, aktive Befehle oder bislang gesammelte Nahrungsmittel (vgl. Abb. 1). Zur besseren Übersichtlichkeit und zur Analyse der erstellten Spielstrategien kann das Spiel unterbrochen werden. Eine optionale Konsole bietet Textausgaben, die im Quellcode zu beliebigen Zeitpunkten erzeugt werden können. Dies ermöglicht dem Studierenden weitere Optimierungsmöglichkeiten in Bezug auf seine Programmierung. Nach der Umsetzung des Spielkonzepts sollen weitere Arbeiten in Richtung einer Client-Server Umgebung für Online-Multiplayerspiele sowie ein Einsatz im Bereich Mobile Gaming starten. Der aktuelle Stand wird im WS 04/05 in der Lehre eingesetzt werden.

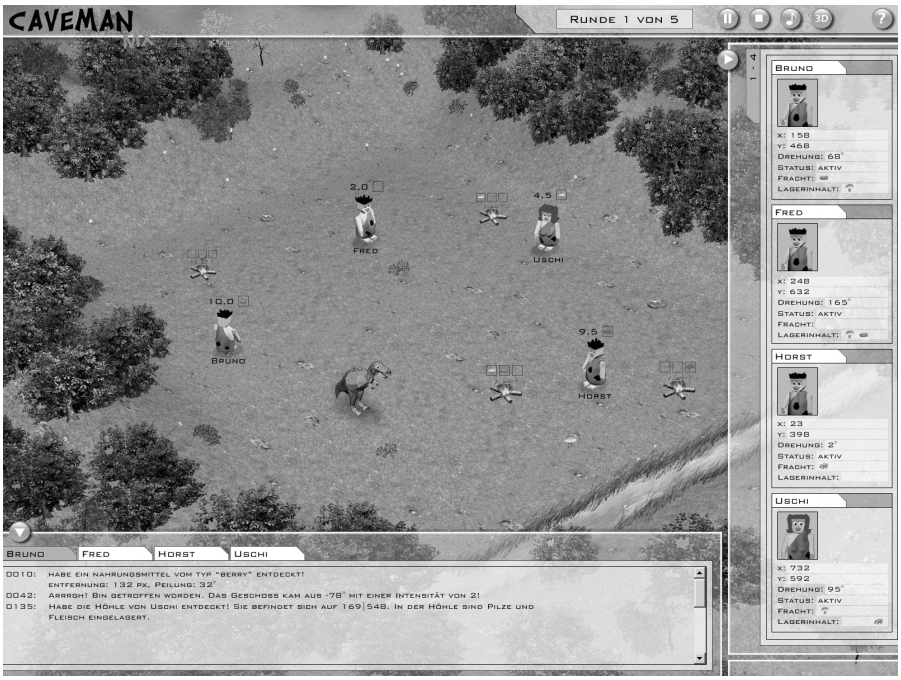

Abb. 1: Screenshot des aktuellen Prototyps

Literaturverzeichnis

[HNR03] Hartmann, W.; Nievergelt, J.; Reichert, R.: Programmieren mit Kara. Springer, Berlin, 2003
[Ka96]] Kahn, K.: ToonTalk – An Animated Programming Environment for Children, Journal Visual Language and Computing, 1996
[CS95] ypher, A.; Smith, D. C.: KidSim: End User Programming of Simulations, CHI, 1995
[CW] http://www.corewars.org/
[RobC] ttp://robocode.alphaworks.ibm.com/home/home.html