Rapid Prototyping von Interaktionskonzepten in der universitären MCI-Lehre

Julian Fietkau, **Martin Christof Kindsmüller**, **Timo Göttel**
Human-Computer Interaction
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
D-22527 Hamburg
{julian.fietkau, mck, timo.goettel}@informatik.uni-hamburg.de

Abstract: In der Lehre zur MCI (Mensch-Computer-Interaktion) stellt sich immer wieder die Herausforderung, praktische Übungen mit spannenden Ergebnissen durchzuführen, die sich dennoch nicht in technischen Details verlieren sondern MCI-fokussiert bleiben. Im Lehrmodul „Interaktionsdesign" an der Universität Hamburg werden von Studierenden innerhalb von drei Wochen prototypische Interaktionskonzepte für das Spiel Neverball entworfen und praktisch umgesetzt. Anders als in den meisten Grundlagenkursen zur MCI werden hier nicht Mock-Ups, sondern lauffähige Software entwickelt. Um dies innerhalb der Projektzeit zu ermöglichen, wurde Neverball um eine TCP-basierte Schnittstelle erweitert. So entfällt die aufwändige Einarbeitung in den Quellcode des Spiels und die Studierenden können sich auf ihre Interaktionsprototypen konzentrieren. Wir beschreiben die Erfahrungen aus der mehrmaligen Durchführung des Projektes und erläutern unser Vorgehen bei der Umsetzung. Die Ergebnisse sollen Lehrende im Bereich MCI unterstützen, ähnliche praxisorientierte Übungen mit Ergebnissen „zum Anfassen" zu gestalten.

1 Einleitung

Das Lehrmodul Interaktionsdesign an der Universität Hamburg hat die Zielsetzung, als Grundlagenmodul für Mensch-Computer-Interaktion im Sinne der Empfehlung der Gesellschaft für Informatik e. V. zu dienen [Fac06]. Übergreifende Lernziele umfassen hierbei u. a. die Grundlagen menschlicher Informationsverarbeitung, Arten von Ein-/Ausgabegeräten und mögliche Interaktionstechniken. Die GI schlägt zur praktischen Vertiefung vor allem Übungen zu Spezifikation und Prototyping vor. In Hamburg wird als darüber hinausgehender Teil der Einführungsveranstaltung im Rahmen eines dreiwöchigen Kurzprojekts ein lauffähiger Software-Prototyp entwickelt. Diese interaktiven Prototypen ermöglichen den Studierenden, neue Interaktionsprinzipien zu explorieren, zu entwerfen und zu evaluieren.

Solche Projekte sind im Rahmen eines MCI-Kurses i. d. R. schwer umzusetzen, weil die Softwareentwicklung nicht Kernthema sein soll und die Teilnehmer bezogen auf das Programmieren unterschiedliche Vorerfahrungen und Vorlieben haben. Damit sich die Studierenden stattdessen möglichst auf ihre Ideen und deren Umsetzung konzentrieren können, soll das verwendete System in Bezug auf die Programmierung leicht und flexibel ansprechbar sein.

Im Jahr 2010 wurde in diesem Rahmen ein fünfwöchiges Mini-Projekt auf Basis des Spiels SheepLifter [Jep10] durchgeführt. SheepLifter ist in Java implementiert, womit die Studierenden in den einführenden softwaretechnischen Veranstaltungen an der Universität Hamburg arbeiten. Die Hoffnung der Organisatoren, dass sich die Studierenden schnell im Quellcode des Spiels zurecht finden würden, erfüllte sich nicht. Es gab über die gesamte Projektlaufzeit große Probleme, u. a. bereits beim Kompilieren des Codes. Die Studierenden mussten sich mehr mit fremdem Code befassen als mit ihren eigenen konzeptuellen Ideen, so dass eigentliche MCI-Inhalte kaum zur Geltung kamen.

Für 2011 musste der Projektzeitraum aufgrund von organisatorischen Umständen auf drei Wochen verkürzt werden. Im Rahmen der Neuplanung wurde als Basis das Spiel Neverball [Rob14] gewählt. Es hat ein einfaches Spielkonzept: Eine Kugel muss durch ein Labyrinth ins Ziel bewegt werden, indem die 3D-Spielwelt in zwei Richtungen geneigt wird. Diese Grundidee bietet viel Spielraum für innovative Interaktionskonzepte mit Eingabe-Hardware aller Art. Gleichzeitig hat das Spiel einen überschaubaren „Funktionsumfang“, der schnell zu erfassen ist und zum Ausprobieren einlädt.

Bei Neverball handelt es sich um freie Software, daher hatten die Lehrenden die Möglichkeit, die Software vor Beginn des Projekts um eine ex-

terne Programmierschnittstelle zu erweitern. Neverball wird normalerweise entweder mit der Maus oder mit den Pfeiltasten der Tastatur gesteuert. Die von uns für das Projekt modifizierte Version besitzt zusätzlich eine Netzwerkschnittstelle (vgl. Abschnitt 2). Das Ziel hierbei ist, eine übersichtliche Schnittstelle zu bieten, durch die das Spiel leicht von eigener Software gesteuert werden kann, ohne dass ein beträchtlicher Teil des internen Programmcodes des Spiels verstanden werden muss. In dieser Hinsicht gibt es Parallelen zu Programmier-Lernumgebungen wie Greenfoot [HK04], welche jedoch im Gegensatz zu unserer Schnittstelle normalerweise auf eine vorher festgelegte Programmiersprache beschränkt sind (bei Greenfoot ist das Java).

Einen Einblick in die Ideenvielfalt der Ergebnisse aus der mehrfachen Durchführung des Projektes geben wir in Abschnitt 3. In Abschnitt 4 findet eine übergreifende Bewertung des Projekts statt.

2 Umsetzung

Im Rahmen des Projektes treffen jedes Jahr Studierende mit heterogener Programmiererfahrung aufeinander. Zwar ist der C-Quellcode von Neverball verfügbar, Vorkenntnisse mit C sind jedoch selten, sodass eine Beschränkung auf C die Situation im Vergleich zu SheepLifter noch verschlechtern würde.

Um den Studierenden mehr Freiraum zu lassen und die unterschiedlichen Vorkenntnisse nutzbar zu machen, wurde Neverball um eine einfache Netzwerkschnittstelle erweitert, welche mit jeder modernen Programmiersprache (neben Java und C also z. B. auch Python, C# und viele andere) genutzt werden kann. Diese spiegelt die internen Variablen des Spiels für die Neigung in x- und z-Richtung sowie die Parameter der Kamera nach außen und bildet so eine vergleichsweise klar definierte und überschaubare Schnittstelle. In der Hauptschleife des Spiels wird kontinuierlich ein Socket eines definierten TCP-Ports abgefragt. Sobald dort ein passendes Datenpaket eintrifft, werden die internen Variablen entsprechend aktualisiert. Der Zusammenhang der verschiedenen Komponenten ist in Abbildung 1 dargestellt.

Zusätzlich zum modifizierten Neverball-Quellcode erhalten die Studierenden Beispielprogramme in C bzw. Java, die die Client-Seite des Protokolls exemplarisch umsetzen und eine Steuerung des Spiels über die Pfeiltasten der Tastatur erlauben. An diesem Code können sich die Studierenden orientieren. Der gesamte Quellcode zum Projekt ist frei lizensiert online verfügbar[1].

1 GitHub: https://github.com/jfietkau/neverball-fbiuhh.

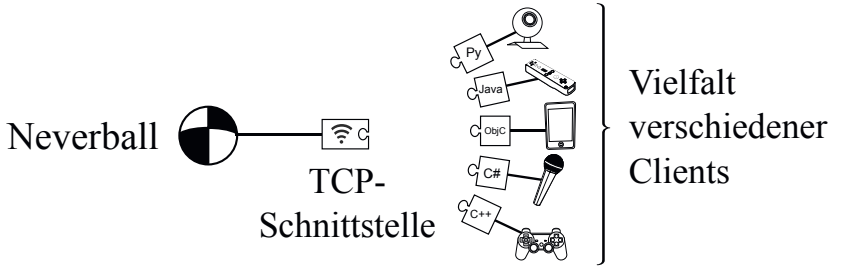

Abbildung 1: Schematische Darstellung der Projekttechnologie. Das Spiel Neverball wurde um eine plattformunabhängige Netzwerkschnittstelle erweitert.

3 Ergebnisse der Durchführung

Das Neverball-Projekt wurde im Lehrmodul Interaktionsdesign in den Sommersemestern 2011, 2012 und 2013 erfolgreich durchgeführt. In allen drei Jahren konnten die Studierenden jeweils innerhalb von drei Wochen eine Vielzahl von Interaktionsprototypen umsetzen. Im Folgenden werden protoypische und besonders innovative Umsetzungen geschildert:

- mehrere Prototypen auf Basis von Bildverarbeitung, teilweise mit der Microsoft Kinect, teilweise mit einer handelsüblichen Webcam und einem Augmented-Reality-Framework;
- spezialisierte Eingabe-Hardware wie das Sensable Phantom, die 3Dconnexion 3D-Maus oder eine Tanzmatte (Abb. 2);
- selbstgebaute Eingabe-Hardware sowohl auf Basis der Arduino-Plattform als auch Lego MindStorms;
- Nutzung der Neigungssensoren in Android-Smartphones, der Nintendo WiiMote sowie dem Apple MacBook Pro;
- Ansteuerung diverser klassischer Gamepads verschiedener Hersteller.

Diese Vielzahl an unterschiedlichen Lösungsansätzen und die Anzahl erfolgreich abgeschlossener Projekte innerhalb des vergleichsweise kurzen Prototyping-Zeitraums deuten darauf hin, dass das Schema einer sehr einfachen netzwerkbasierten Schnittstelle geeignet ist, die Entwicklung verschiedenster Interaktionsprototypen in kurzen Zeiträumen zu ermöglichen. Anders als im SheepLifter-Projekt konnten die Teilnehmer ihre Zeit und ihre mentale Energie in die Planung und Umsetzung ihrer Entwurfsideen investieren, statt sich in fremde Technik einzuarbeiten. Dennoch blieb die Möglichkeit der in-

tensiven Programmierarbeit für jene Teams erhalten, die diese gerne nutzen wollten.

Abbildung 2: Neverball-Spieler auf einer Tanzmatte.
Der Fuß auf dem rechten Pfeil kippt die Spielwelt nach rechts.

Beispielhaft lassen sich zwei Prototypen aus dem Jahr 2012 gegenüberstellen, die beide auf der *Microsoft Kinect* (ein zur Erkennung von Gesten geeignetes Kamerasystem) basierten. Das eine Team griff auf vorhandene Frameworks und Bibliotheken für Gestenerkennung zurück und erreichte dadurch Planungsfreiheit auf einem eher hohen Abstraktionsniveau, ohne sich mit vielen technischen Details befassen zu müssen. Das andere Team implementierte eine eigene Gestenverarbeitung auf Basis von OpenCV, welche zwar weniger Robustheit und Entwurfsflexibilität erreichte, aber dafür im Detail exakt an die Projektanforderungen angepasst werden konnte. Beide Teams setzten unterschiedliche Schwerpunkte für ihren Entwurfsraum, erreichten jedoch auf ihren jeweiligen Wegen in beiden Fällen eine hohe Ergebnisqualität.

4 Bewertung

Die Parallelen und Unterschiede in der Durchführung zwischen den Projekten basierend auf SheepLifter 2010 und Neverball 2011 bis 2013 erlauben einen ersten – wenn auch eher explorativ anekdotischen – Vergleich. Im Neverball- Projekt waren die Studierenden nicht gezwungen, sich in eine bestehende Code-Basis einzuarbeiten. Stattdessen gab es eine simple und leicht zu erklärende Schnittstelle. Das Resultat war eine höhere Qualität der Projekt-Ergebnisse trotz kürzerem Entwicklungszeitraum: Der Anteil der vollendeten Projekte (d. h. Erreichen des selbstgesetzten Ziels) war deutlich höher, die

Umsetzungen konnten intensiver getestet und verbessert werden. Gleichzeitig sank der Betreuungsaufwand.

Hinsichtlich der Lernziele erreicht unser Ansatz einen tiefergehenden Umgang mit möglichen Interaktionstechnologien und -konzepten als das vorherige Projekt. Die praktische Anwendung des Wissens in einem selbstgestalteten Kontext erleichtert das nachhaltige Lernen [Pap93, RMSS96].

Für diese Anpassung der Software um eine solche Schnittstelle ist natürlich auch eine Einarbeitung in den Quellcode unabdingbar, allerdings nur einmal und auf Seiten der Veranstalter. Im Fall von Neverball konnte ein erfahrener Student dies in weniger als zwei Arbeitstagen umsetzen.

Eine in jedem Jahr erkennbare Schwäche des Projekts ist, dass einige naheliegende Steuerungskonzepte mehrfach und wiederholt aufgegriffen wurden. Bezogen auf Neverball entwickelten jedes Jahr mehrere Projektteams Konzepte, in denen die Neigung der Spielwelt direkt auf die Neigungssensoren eines Smartphones abgebildet wurde. Obwohl es unser Ziel war, innovative Ideen zu fördern, war festzustellen, dass einige Teams sich bereits mit sehr naheliegenden Umsetzungen zufrieden stellten, dafür aber dort sehr fokussiert waren, um diese Idee gut umzusetzen.

Die aktuelle technische Umsetzung der Neverball-Schnittstelle lässt noch Raum zur Erweiterung. Beispielsweise wäre eine gleichzeitige Anbindung mehrerer Clients technisch denkbar, ebenso wie ein zweiter Informationsfluss von Neverball zum Client, um Ereignisse im Spiel (wie etwa die Kollision mit einer Wand) im Eingabegerät verarbeiten und darauf reagieren zu können. Diese Erweiterungen wurden bisher nicht hoch priorisiert, wären aber mit angemessenem Aufwand umsetzbar.

5 Fazit

Lehrende im Bereich MCI haben die schwierige Aufgabe, den Bereich zwischen reinen „Lo-Fi“-Übungen und ausgewachsenen Softwareprojekten zu treffen, in dem einerseits etwas entwickelt wird, was später tatsächlich ausprobiert werden kann, und andererseits die techniknahe Programmierung nicht das Kernthema verwässert. Es hat sich bei uns gezeigt, dass das Konzept für die Übungsprojekte im Interaktionsdesign-Modul in dieser Hinsicht das richtige Abstraktionsniveau trifft und sowohl für die Studierenden als auch für die Lehrenden eine große Bereicherung darstellt. Anders als die Projekte aus den Jahren zuvor funktioniert das Neverball-Projekt auch in dem kurzen

Zeitraum über alle Übungsgruppen und Semester hinweg sehr gut. Wir hoffen, dass diese Idee auch die MCI-Lehre an anderen Orten inspirieren kann.

Literaturverzeichnis

[Fac06] Fachgruppe Software-Ergonomie. *Curriculum für ein Basismodul zur Mensch-Computer-Interaktion*. Gesellschaft für Informatik e. V. (GI), Bonn, 2006.

[HK04] Poul Henriksen und Michael Kölling. Greenfoot: Combining Object 7 Visualisation with Interaction. In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '04, S. 73–82, New York, NY, USA, 2004. ACM.

[Jep10] Jeppe Schmidt et al. SheepLifter. https://code.google.com/p/sheeplifter/, 2009–2010.

[Pap93] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, 2nd ed. Auflage, 1993.

[RMSS96] Mitchel Resnick, Fred Martin, Randy Sargent und Brian Silverman. Programmable Bricks: Toys to Think With. In IBM *Systems Journal*, Jgg. 35, S. 443–452, 1996.

[Rob14] Robert Kooima et al. Neverball. http://neverball.org/, 2001–2014.