Konzepte zur „begreifbaren Lehre“ in der Regensburger Medieninformatik

Manuel Burghardt, Markus Heckner, Tim Schneidermeier, Felix Raab, Christian Wolff

Lehrstuhl für Medieninformatik, Universität Regensburg

Zusammenfassung

In diesem Beitrag stellt die Regensburger Medieninformatik unterschiedliche Ansatzpunkte für begreifbare Lehre vor, welche sich im Rahmen des Medieninformatik-Curriculums ergeben.

1 Rahmenbedingungen

Medieninformatik kann seit dem Wintersemester 2010 / 2011 auch in Regensburg studiert werden. Derzeit ist ein B.A.-Studium der Medieninformatik in Verbindung mit mindestens einem zweiten, frei wählbaren Fach möglich, ein einzügiger, vierjähriger B. Sc. befindet sich derzeit in der Planung. Die Regensburger Medieninformatik ist Teil des *Instituts für Information und Medien, Sprache und Kultur*, und versteht sich dabei als eine angewandte Informatik, für die sich aus den benachbarten Disziplinen (z. B. Informationswissenschaft, Kulturwissenschaft, Medienwissenschaft, etc.) des Instituts vielfältige Anwendungsszenarien und Aufgabenstellungen ergeben. Aus dieser interdisziplinären Ausrichtung ergibt sich auch der Bedarf an innovativen und über die Fachgrenzen hinaus *begreifbaren* Lehrkonzepte. Als eine angewandte Informatik lehrt und erforscht die Medieninformatik innovative Interaktionsformen), mobile interaktive Systeme sowie *usability* und *media engineering*, und setzt sich dabei vor allem mit praxisorientierten Fragestellungen, wie etwa der nutzerzentrierten Systementwicklung, auseinander (Wolff, 2009). Die erforderliche Theorie wird dabei stets im Kontext praktischer Systementwicklung vermittelt.

2 Konzepte zur „begreifbaren Lehre“

In diesem Abschnitt werden ausgewählte Konzepte vorgestellt, die helfen sollen, einem interdisziplinären und breitgefächerten Publikum praxisorientierte Medieninformatikinhalte zu vermitteln und *begreifbar* zu machen.

2.1 Konstruktiver Umgang mit neuen Interaktionstechniken

Interaktive und multimodale Systeme durchdringen den Alltag in nahezu allen Lebensbereichen (*ubiquitous media and computing*). Einen besonderen Stellenwert in Forschung und Lehre der Regensburger Medieninformatik nimmt deshalb das Thema *Human Computer Interaction (HCI)* ein. Um neue Interaktionstechniken (z.B. Gesten- oder Blicksteuerung) verständlich(er) zu machen, wird nicht nur auf "professionelle" Hardware gesetzt, sondern auch auf technisch weniger komplexe Eigenbauvarianten. Diese selbst gefertigten *low cost*-Interaktionsgeräte verwenden einfache, preisgünstige Komponenten sowie *open source*-Software und vermitteln anschaulich jene Interaktionstechniken, welche auch in der professionellen Hardware zu finden sind. Die Studenten können anhand der *low cost*-Varianten die grundlegende Funktionsweise der Geräte und die dahinter stehende Interaktionstechnik auf einem einfacheren Niveau erfahren. Beispielhaft wurden für den Bereich der Blick- und Gestensteuerung auf Basis von *open source*-Software und frei verfügbaren Tutorials bisher insgesamt drei unterschiedliche *low cost*-Varianten gefertigt, die im Medieninformatiklabor jeweils eine Entsprechung in Form professioneller Geräte finden (vgl. Abbildung 1).

Es sind dies (1) ein mobiler Eyetracker[1], der die Erfassung und Auswertung von Blickrichtung und Blickbewegung des Benutzers erlaubt, und damit eine weitere Modalität für die HCI ermöglicht, (2) ein interaktives Whiteboard[2], welches herkömmliche, analoge Boards um die digitale Ebene erweitert und damit sowohl das Speichern von Notizen als auch die Touch-Interaktion mit Objekten auf dem Board erlaubt sowie (3) ein Multitouch-Tisch[3], dessen Touchscreen mehrere Eingaben gleichzeitig wahrnehmen und verarbeiten kann.

2.2 Interaktionsprogrammierung begreifbar machen

Für die Medieninformatik ist die Vermittlung von Kenntnissen in der Anwendungs- und Interaktionsprogrammierung ein Kernaspekt der Lehre. Grundsätzlich geht es dabei um die Erweiterung der systematischen Grundkenntnisse, wie sie in einführenden Programmierkursen vermittelt werden, mit Bezug auf konkrete Aufgabenstellungen der Anwendungsprogrammierung (z. B. Gestaltung von Benutzerschnittstellen, Ein-/Ausgabeprogrammierung, Datenbankanbindung etc.). Die folgenden Lehrkonzepte illustrieren die in Regensburg verwendeten Ansätze zur Vermittlung *begreifbarer* Interaktionsprogrammierung.

---

1 Für eine Anleitung zum Bau des low cost-Eyetrackers (Kraus, 2010) siehe *http://igazeweb.blogspot.com/*, der high end-Eyetracker stammt von der Firma SMI (vgl. *http://www.smivision.com/*)

2 Anleitung zum Bau des low cost-Whiteboard: *http://johnnylee.net/projects/wii/*, *das high end-Whiteboard* ist ein Hitachi StarBoard (vgl. *http://www.hitachisolutions-eu.com/de/products/interactive_whiteboards/index.php*)

3 Anleitung zum Bau des low cost-Multitouch Pads siehe *http://www.anneroudaut.fr/fun/acrylicpad.html*), der high end-Multitouch Tisch stammt von der Firma Evoluce (vgl. *http://www.evoluce.com/de/index.php*)

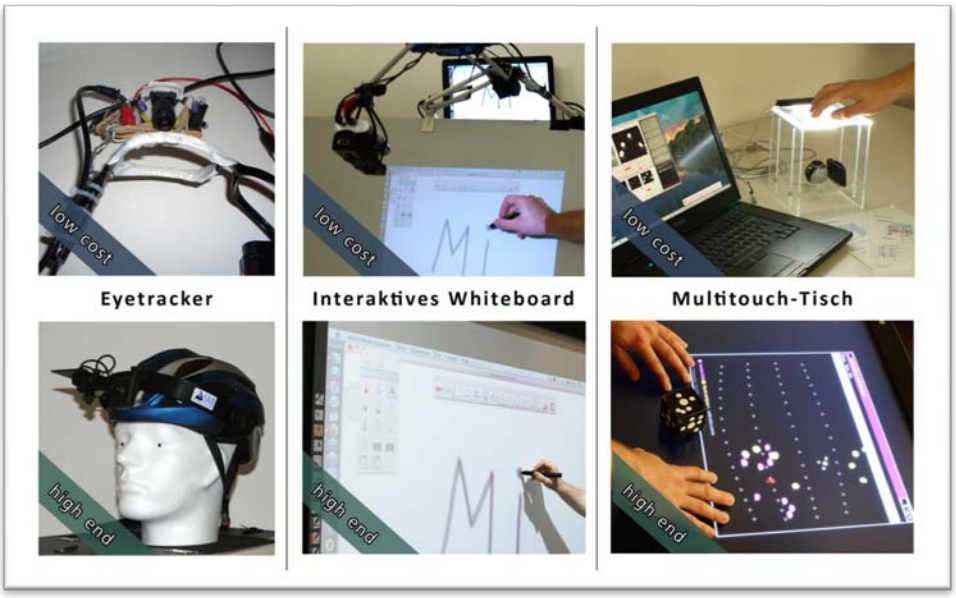

Abbildung 1: Die Abbildung zeigt oben jeweils die low cost-Variante des Eyetrackers, des interaktiven Whiteboards und des Multitouch-Tisches, und unten die high end-Entsprechung, die sich im Medieninformatiklabor findet.

2.2.1 Entwicklung interaktiver Systeme auf der Basis von *Android*

Zu den grundlegenden Konzepten, die für die Entwicklung interaktiver Systeme verstanden werden müssen, gehören *Events*, *asynchrone Aufrufe* von Methoden über *Callbacks*, nebenläufige Verarbeitung mit *Threads* und der Aufbau der Systemarchitektur nach dem MVC-Prinzip (*Model*, *View*, *Controller*). Diese Konzepte werden in einer Veranstaltung im Umfang von vier Semesterwochenstunden (2S + 2Ü) am Beispiel der Plattform *Android* (Becker, 2010; Meier, 2010) vorgestellt und in einer begleitenden Übung direkt von den Studenten umgesetzt und vertieft. Android ermöglicht einerseits schnelle Resultate und bietet zum anderen ein komplettes Framework, anhand dessen sich alle Prinzipien der Interaktionsprogrammierung demonstrieren lassen. Beispielsweise erzwingt die Architektur des Android-Frameworks die MVC-Aufteilung, da *Controller* und *View* klar konzeptuell getrennt sind. Somit werden *Software Design Patterns* (Gamma, 2008) schon zu Beginn des Studiums vermittelt und guter Programmierstil gefördert. Da Android-Apps in Java entwickelt werden, kann dieser Kurs direkt auf der Grundlagenvorlesung zur objektorientierten Programmierung in Java aufbauen. Ferner lernen die Studenten sehr früh mit API-Dokumentationen umzugehen und sich selbständig in Programmierframeworks einzuarbeiten, was den späteren Umstieg auf weitere Frameworks stark erleichtert. Zusätzlich begünstigt die Verwendung von Java aufgrund der weiten Verbreitung den Wissenstransfer auf andere Felder der Anwendungsprogrammierung, da sich die für Android gelernten Software Patterns auch in anderen Domänen wiederfinden. Hinsichtlich der Motivation der Studenten bietet die Programmierung mit Android einen weiteren entscheidenden Vorteil: Über den Android Market können

die im Kurs entwickelten Apps schnell und unkompliziert potenziellen Benutzern präsentiert werden (vgl. z. B. http://www.small-worlds.de/presenter/), d. h. man entwickelt die Software nicht zum reinen Selbstzweck (d. h. für den Dozenten), sondern für eine große Zahl real existierender Android-Nutzer.

2.2.2 Programmieren mit *Processing*

Neben der praxisnahen Vermittlung von objekt-orientierter Programmierung und Framework-Nutzung mit Android wird auch *Processing* (Reas & Fry, 2007) in ergänzenden Übungen eingesetzt. Ursprünglich entwickelten Studenten am MIT die quelloffene, java-basierte Programmierumgebung um grundlegende Konzepte der Informatik in visuellem Kontext zu erklären, inzwischen erfreut sich das Tool aber auch bei Designern und Künstlern großer Beliebtheit (vgl. http://www.processing.org/about/). Da sich mit Processing sehr schnell ansprechende Visualisierungen und Interaktionen umsetzen lassen, eignet sich das Werkzeug hervorragend dazu abstrakte Prinzipien der Programmierung anschaulich und begreifbar zu vermitteln. Auf diese Weise können Algorithmen, Datenstrukturen und objekt-orientierte Zusammenhänge an nachvollziehbaren Beispielen mit direktem visuellem Feedback gelehrt werden. Beispielsweise erfolgte im Rahmen einer Übung die Vermittlung grundlegender Programmierkonzepte mit Processing zunächst an PCs mit Hilfe von Emulatoren, anschließend konnten die Teilnehmer ihre Multitouch-Applikationen auf dem Interaktionstisch (s.o.) des Medieninformatik-Labors gemeinsam ausprobieren und damit die Ergebnisse ihrer Arbeit auch auf professionellen Geräten testen. Medieninformatik wird durch Werkzeuge wie Processing lebendig und greifbar: Im Sinne des top-down-Ansatzes, wie ihn Schweitzer, Boleng, & Graham (2010) vorschlagen, stellt die Vermittlung den *funktionalen Aspekt* in den Mittelpunkt, weniger die algorithmischen und technischen Grundlagen.

2.2.3 *Physical computing* mit *Arduino*

Praxisnahes Begreifen von Medieninformatik ermöglicht auch die open-source-Prototyping-Plattform *Arduino*, mit deren Hilfe Microcontroller programmiert werden können. Ähnlich wie *Processing* bietet Arduino eine leicht erlernbare Programmierumgebung, die stark von der Hardware abstrahiert, der entwickelte Code ist dadurch leicht lesbar und gut weiterzuentwickeln. Die Studenten können mit der *physical computing*-Plattform erste Hardware-Prototypen entwickeln und somit frühzeitig testen, ob ihre Ideen tragfähig sind (Brock, Bruce, & Reiser, 2009; Oxer & Blemings, 2009).

2.2.4 (Be-)Greifbare User Interfaces durch *Prototyping*

Die Entwicklung von *User Interfaces* (UI) beginnt nicht mit Coding, sondern kann erst erfolgen, nachdem die Anforderungen der Benutzer klar verstanden worden sind. Ein einfach zu vermittelndes und durch seine Anschaulichkeit sehr produktives Mittel für solche Anforderungsanalysen ist das Entwickeln und Testen von Papierprototypen (Snyder, 2007) mit „echten“ Benutzern. Ein UI anhand eines Papierprototypen zu konzipieren und zu entwickeln schafft bei den Studenten ein grundlegendes Problembewusstsein und kann ihnen sprichwörtlich die Augen öffnen. Gleichzeitig ist das Verfahren schnell und direkt im Seminarraum durchführbar, da die zeitaufwändige Entwicklung funktionaler Prototypen oder auch nur mit Editoren erzeugter Bildschirm-mock-ups entfällt, und dafür auch keine technische Infrastruk-

tur erforderlich ist. Der Einsatz von *paper prototyping* (und anderer Entwicklungsmethoden) ist daher ein wichtiger Teil des Curriculums und in unterschiedliche Lehrformen integriert, u. a. durch aktivierende und interaktive Seminare sowie bei teambasierten Projektseminaren.

3 Ausblick

Der Beitrag illustriert verschiedene Ansätze für *begreifbare* Vermittlung medieninformatischer Inhalte. Abschließend kann man argumentieren, dass die *Medialität* des Gegenstandsbereichs der Medieninformatik im Allgemeinen (Mensch-Maschine-Schnittstelle) sowie im Besonderen (*ubiquitous computing*, *tangible interaction*, *etc.*) *Begreifbarkeit* in doppelter Hinsicht als zentrales Konzept erscheinen lässt: Einerseits Begreifbarkeit als Ziel von Vermittlungsprozessen (didaktischer Aspekt), andererseits Begreifbarkeit als Aspekt des Erlebens der Interaktion mit physischen und / oder virtuellen Artefakten.

Literatur

Becker, A. (2010). Android 2. Heidelberg: dpunkt.

Brock, J. D., Bruce, R. F., & Reiser, S. L. (2009). Using Arduino for introductory programming courses. J. Comput. Small Coll., 25(2), 129–130. USA: Consortium for Computing Sciences in Colleges.

Gamma, E. (2008). Design patterns. Boston [u.a.]: Addison-Wesley.

Kraus, D. (2010). Web Browsing Using Head and Gaze Control. Basic Interaction Realized with Low-Cost Eye and Head Tracking Systems. Universität Regensburg. http://igazeweb.blogspot.com, letzter Zugriff am 28.06.2011

Meier, R. (2010). Professional Android 2 application development. Indianapolis, Ind. Wiley.

Oxer, J., & Blemings, H. (2009). Practical Arduino: Cool Projects for Open Source Hardware. Berkeley/CA: Apress.

Reas, C., & Fry, B. (2007). Processing: A Programming Handbook for Visual Designers and Artists (p. 736). The MIT Press.

Schweitzer, D., Boleng, J., & Graham, P. (2010). Teaching introductory computer graphics with the processing language. Journal of Computing Sciences in Colleges, 26(2), 73–79. USA: Consortium for Computing Sciences in Colleges.

Snyder, C. (2007). Paper prototyping : the fast and easy way to design and refine user interfaces. Amsterdam [u.a.]: Morgan Kaufmann.

Wolff, C. (2009). „embedded media computing“ – die Regensburger Ausrichtung der Medieninformatik. (M. Herczeg, Ed.). Berlin, Humboldt-Universität: Online-Plattform der GI-Fachgruppe Medieninformatik auf uni.commsy.net.