jugend forscht

Bundeswettbewerb 2007 – Arbeitswelt

Neue Möglichkeiten der Programmbedienung durch einen Mehrfinger-Touchscreen

Pascal Schmitt

Bischlingstr. 2
63846 Laufach
☎ 06093/994871
pascal@germroth.name
⋆ 15. Februar 1988
Klasse: K12

Hanns-Seidel-Gymnasium Hösbach

An der Maas 2
63768 Hösbach
☎ 06021/44989-0

Projektbetreuer: Claus-Peter Horn, OStR

Der hohe Verbreitungsgrad der Computermaus lässt die Anwender oft vergessen, dass sie keineswegs das optimale Eingabegerät darstellt: Besonders unerfahrenen Benutzern fällt es schwer, von der horizontalen Mausebene auf die vertikale Bildschirmebene umzudenken. Wesentlich effizienter arbeitet man mit Touchscreens, bei denen das Ziel direkt berührt werden kann. Allerdings unterlagen diese noch bis vor kurzem der Einschränkung, nur einen einzigen Berührpunkt verarbeiten zu können und stellten diesbezüglich keine Verbesserung zur Computermaus dar.
Im Rahmen dieser Arbeit wurde ein neuartiger, kostengünstiger Touchscreen gebaut – sowie die dazugehörige quelloffene Software entwickelt – welcher im Gegensatz zu den bekannten Modellen gleichzeitig die Position beliebig vieler Finger erfassen kann und dadurch eine wesentlich natürlichere Interaktion mit dem Computer erlaubt.
Diese Arbeit zeigt die neuartigen Möglichkeiten für die Gestaltung von Benutzeroberflächen auf und demonstriert an selbstentwickelten Beispielprogrammen die vielfältigen Vorteile dieser Technik.

Hösbach, 6. April 2007

Inhaltsverzeichnis

1 Problemstellung und Zielsetzung

Seit der Erfindung der Computermaus 1964 und ihrer anschließenden Etablierung veränderten sich die Bediengeräte von Computern nicht mehr: Zwar werden stetig neue Technologien entwickelt, doch keine erzielte bislang den Verbreitungsgrad der Maus.

Dabei ist diese keineswegs das optimale Eingabegerät. Insbesondere die Steuerung des Mauszeigers in der vertikalen Bildschirmebene durch die Bewegung der Maus auf der horizontalen Tischebene sorgt bekanntlich für Probleme, hauptsächlich bei unerfahrenen Nutzern. Der Touchscreen stellt die natürlichste Lösung dieses Problemes dar, denn er erlaubt, direkt mit der Benutzeroberfläche zu interagieren statt indirekt über den Mauszeiger. Der hohe Verbreitungsgrad, besonders an öffentlich genutzten Terminals wie etwa Fahrkartenautomaten, zeigt, dass im Gegensatz zur Maus oder den kurze Zeit aktuellen Trackballs, Touchscreens wohl die größte Akzeptanz bei den Benutzern genießen.

Allerdings haben aktuelle Touchscreens folgendes Grundproblem mit der Maus gemeinsam: Die Interaktion erfolgt nur mit einer Hand bzw. einem einzigen Finger, was unnatürlich, ineffizient und, da der Touchscreen aussschließlich mit dem Finger berührt werden darf, auf Grund der Armhaltung auch ermüdend ist.

Daher stellt sich die Frage, wie herkömmliche Touchscreens so verbessert werden können, dass mit ihnen eine natürlichere, dem Verhalten in der „realen Welt“ angepasste Interaktion ermöglicht wird.

Ziel dieser Arbeit war, einen neuartigen Mehrfinger-Touchscreen zu entwickeln, der zur Lösung der angesprochenen Probleme beitragen kann. Darüber hinaus war jedoch der Schwerpunkt dieser Arbeit, die neuen Möglichkeiten für die Gestaltung von Benutzeroberflächen und der Programmbedienung an Hand selbst entwickelter Beispielprogramme zu demonstrieren.

Da die verwendete Technik auf einen physikalischen Effekt beruht, könnte diese Arbeit im Wettbewerb „Jugend forscht“ in den Fachbereich „Physik“ eingeordnet werden. Allerdings wurde auch ein solcher neuartiger Touchscreen gebaut, was eine Teilnahme im Fachbereich „Technik“ rechtfertigen würde. Außerdem besteht ein nicht geringer Teil der Arbeit aus Auswertungs- und Demonstrationssoftware, was eher „Informatik“ entspricht. Die Wahl von „Arbeitswelt“ als Fachbereich dieser Arbeit wurde aus dem Grund getroffen, dass die hier vorgestellte „Mehrfinger“-Bedientechnik enormes Potential für die Verbesserung der Benutzerfreundlichkeit von Computern birgt und folglich in sehr vielen Bereichen der Arbeitswelt gewinnbringend angewendet werden kann.

2 Der Touchscreen

2.1 Nachteile bekannter Touchscreen-Techniken

Der Hauptunterschied zwischen Maus und Touchscreen liegt darin, dass mit der Maus lediglich über eine Geschwindigkeitsmessung indirekt die Zeigerposition verändert wird. Dabei kann Verwirrung verursachen, dass durch eine Verschiebung der Maus z.B. um 1cm je nach Geschwindigkeit der Zeiger um 20Pixel oder die gesamte Bildschirmbreite verschoben wird. Im Gegensatz dazu arbeiten Touchscreens mit absoluten Koordinaten, was natürlicher und einfacher für unerfahrene Benutzer ist.

Alle herkömmlichen Touchscreens haben gemeinsam, dass sie nur je *eine* Messung in x- und y-Richtung ausführen und dadurch nur *genau einen* Berührpunkt errechnen können. Wird die Oberfläche z.B. mit zwei Fingern berührt, verhalten sich die Geräte fehlerhaft: meistens wird dann die Koordinate der Mitte der Druckpunkte ausgegeben.

Da die Bedienung eines Touchscreens mit mehreren Fingern und Händen naheliegend ist, wurden bereits einige Forschungen in diesem Gebiet angestellt: Sony [Rek02] und Mitsubishi [DL01] entwickelten z.B. mit „Smartskin“ und „Diamond Touch“ Sensorflächen, die nicht nur mehrere Druckpunkte unterstützen, sondern auch den angewendeten Druck messen können. Allerdings sind diese Flächen (noch) nicht transparent, sondern das Bild wird von oben auf die Fläche (und die Hände des Benutzers) projiziert, was durch Schatten eine angenehme Arbeitsposition verhindert, da der Raum über der Fläche frei gehalten werden muss.

Eine Alternative, die dieses Problem nicht mit sich bringt und zusätzlich noch mit geringerem Hardwareaufwand auskommt, stellen rein optische Systeme dar: Dabei filmen eine oder mehrere Kameras die Hände der Benutzer vor einem Bildschirm und versuchen die Handpositionen zu errechnen. Da diese „gestenbasierten“ Systeme allerdings noch nicht die erforderliche Genauigkeit aufweisen, die für den Betrieb als Touchscreen nötig wäre, arbeiten sie meist wie Mäuse nur mit relativen Koordinaten.

Das Microsoft-Forschungsprojekt „Touch Light“ [Wil04] arbeitet mit einer „holographischen“ Rückprojektionsfläche, die von hinten aus einem bestimmten Winkel bestrahlt als Rückprojektionsfläche wirkt, sonst aber transparent wie eine Plexiglasscheibe ist. Dies erlaubt, mehrere Kameras direkt hinter der Scheibe anzuordnen, die mit wesentlich höherer Genauigkeit arbeiten und absolute Koordinatenangaben ermöglichen. Der große Nachteil des Systems besteht allerdings darin, dass die verwendeten Projektionsscheiben und Kameras für den privaten Gebrauch deutlich zu teuer sind.

2.2 FTIR-Touchscreens

2.2.1 Aufbau und Funktionsweise

(a) (b)

Abb. 1: Beispiel eines FTIR-Scanners: „Biometrika HiScan“

Die Grundlage des hier vorgestellten Touchscreens bildet die bereits in Fingerabdruckscannern (siehe Abb. 1(a)) eingesetzte Technik „FTIR“ – „frustrated total internal reflection“. Die Idee dazu hatte der New Yorker Professor Jefferson Han [Han05] der auch den ersten derartigen Touchscreen baute, die drei Präsentationsvideos [Han] wurden im Web begeistert kommentiert. Mittlerweile wurde das Spin-Off „Perceptive Pixel“ mit dem Ziel, den ersten Mehrfinger-Touchscreen auf dem Mark zu bringen gegründet, allerdings beschränkt sich die frei verfügbare technische Beschreibung auf ein Bild, das Abb. 2(d) ähnlich ist. Ausgehend von diesem Bild wurde im Rahmen dieser Arbeit eigenständig ein Touchscreen und die zugehörige Software entwickelt.

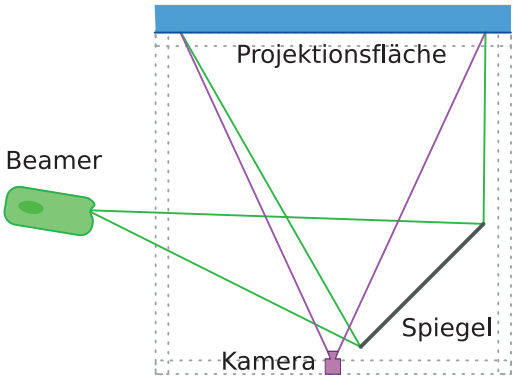
Der Touchscreen (siehe Abb. 2) besteht aus einer horizontal auf einem Würfel aus Aluminium-Profilen in 1m Höhe positionierten Rückprojektionsscheibe, auf welche der Bildschirminhalt mit Hilfe eines Beamers projiziert wird. Die Berührungserkennung erfolgt durch eine Infrarotkamera, die von unten auf die Projektionsfläche gerichtet ist. Wesentliches Funktionsprinzip dabei ist, dass der Beamer und die Kamera in verschiedenen Frequenzbereichen des elektromagnetischen Spektrums arbeiten: das Bild des Beamers ist für die Infrarotkamera unsichtbar. Die Kamera registriert nur das Leuchten der Finger im Infrarotbereich, welches durch Berührung der Oberfläche des Touchscreens entsteht. Beim Berühren der Platte dringt nämlich Infrarotlicht, das von den Kanten her über LEDs in die Platte eingestrahlt wird, in die Finger ein.

Für den Aufbau wurde ein Beamer-Modell gewählt, welches kein Infrarotlicht aussendet.

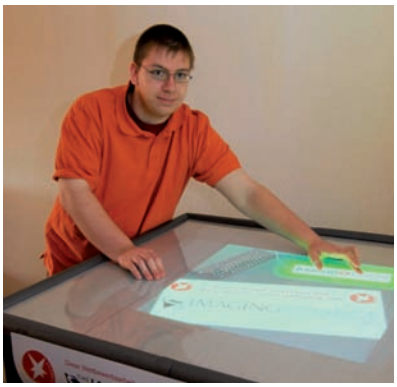
Bei der Infrarotkamera handelt es sich um das Modell DMK 21AF04 (IEEE1394, Monochrom, 640px ∗ 480px) das dankenswerterweise vom der Firma The Imaging Source Europe kostenlos zur Verfügung gestellt wurde.

Die Rückprojektionsfläche besteht aus zwei je $1m^2$ großen Plexiglasplatten zwischen denen Transparentpapier als Mattscheibe eingebracht wird.

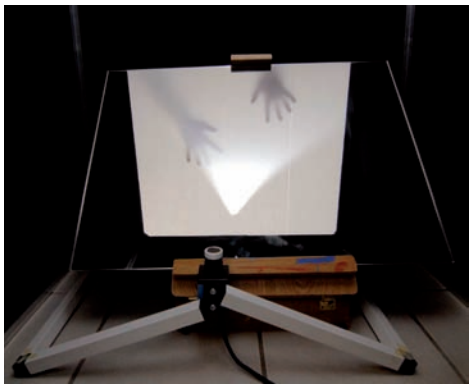
Die Infrarotstrahlung wird von den Kanten her mit 48 Infrarot-LEDs (Osram Power TOPLED SFH 4250, 850nm) in die obere Platte eingebracht. Es wurden abgeflachte Modelle gewählt, die sich besonders einfach an den Kanten anbringen lassen.



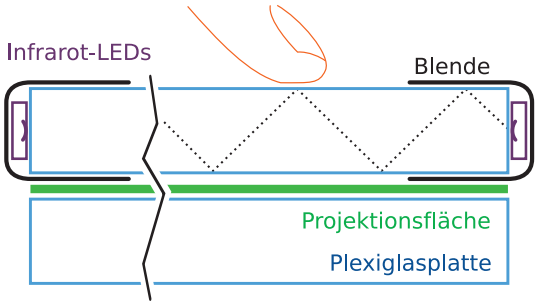
(a) Schema des Aufbaus in Seitenansicht



(b) Touchscreen mit Benutzer



(c) Aufnahme aus Richtung des Beamers mit IR-Kamera im Vordergrund und sich spiegelnder Projektionsfläche



(d) Schematischer Aufbau der Bildschirmoberfläche

Abb. 2: Aufbau des Touchscreens

2.2.2 Physikalische Grundlagen

Der Weg des Lichtes von den LEDs bis in die Finger der Benutzer wird im Folgenden mit Hilfe einfacher Schulphysik erklärt, wobei nur das Phänomen der Totalreflexion von Licht beim Übergang von einem Medium in ein anderes als bekannt vorausgesetzt wird (siehe Abb. 3).

Der Grenzwinkel der Totalreflexion an der Grenzfläche zwischen Platte und Luft beträgt

$$arcsin\frac{n_{\text{Luft}}}{n_{\text{Platte}}} = 39°$$

Folglich tritt nur ein kleiner Teil des Lichtes in der Nähe der Kanten aus und wird an der Blende absorbiert. Der Rest wird total reflektiert und „füllt“ die Platte.

Wird nun allerdings ein Finger auf die Platte gelegt, so bildet sich dort eine Grenzfläche zwischen Platte und oberster Hautschicht an welcher der Grenzwinkel nunmehr

$$arcsin\frac{n_{\text{Haut}}}{n_{\text{Platte}}} = 66°$$

beträgt. Dies bedeutet, dass ein Teil des ansonsten total reflektierten Lichtes an der Stelle der Berührung aus der Platte austritt und den Finger ausschließlich an der berührten Stelle beleuchtet. Dieser reflektiert das Licht durch die Platte und die Projektionsfläche wiederum in die Kamera, die im Inneren des Würfels positioniert ist.

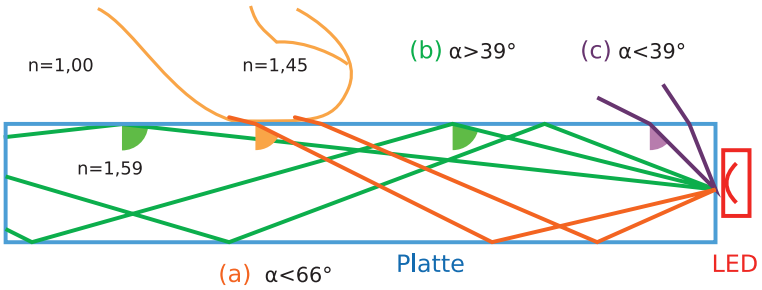


Abb. 3: Die drei betrachteten Fälle: (a) Licht tritt an der Grenzfläche zum Finger aus, (b) Licht wird total reflektiert, (c) Licht tritt aus

Damit ist ein wichtiges Problem gelöst: andere Systeme müssen auf komplizierte, fehleranfällige und ungenaue Weise mit Hilfe mehrerer Kameras Tiefenbilder erzeugen um festzustellen, ob eine Berührung vorliegt. Hier allerdings reicht eine Kamera aus, denn das Licht verlässt die Platte nur, wenn auch wirklich eine Berührung vorliegt. Dies führt auch zur guten Skalierbarkeit des Systems: die Präzision kann einfach erhöht werden, indem eine noch höher auflösende Kamera installiert wird und für ausreichende Beleuchtung gesorgt ist, sonstige Beschränkungen bestehen nicht.

Im folgenden Abschnitt werden die Grundlagen der entwickelten Software erläutert. Die Inhalte sind jedoch nicht Voraussetzung für das Verständnis der darauf folgenden Kapitel.

2.2.3 Grundlagen der Auswertungssoftware

Die verwendete, komplett in C++ unter Verwendung der Bildverarbeitungs-Bibliothek OpenCV (`http://opencvlibrary.sf.net/`) selbstgeschriebene Software zur Registrierung und Weiterverarbeitung der erzeugten Lichtflecken lässt sich in mehrere „Ebenen“ aufteilen. Diese werden im Folgenden in ihrer logischen Reihenfolge beschrieben.

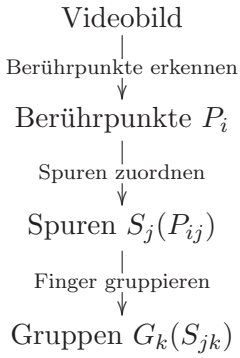
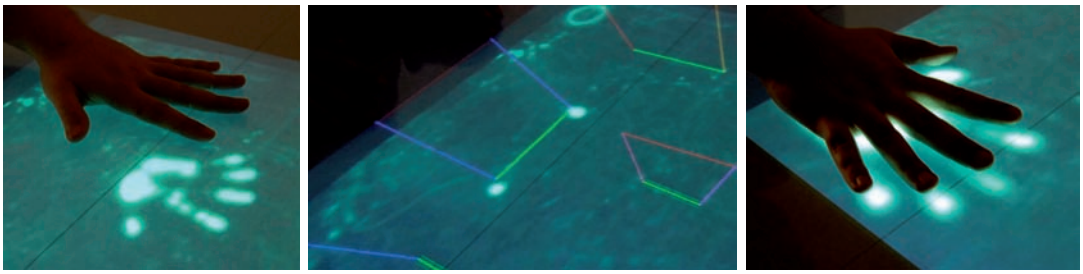


Abb. 4: Schema der Bildverarbeitung

Berührpunkte erkennen Der Touchscreen verwendet eine digitale Graustufen-Kamera, die per IEEE1394 („Firewire“) mit dem Auswertungscomputer verbunden ist. Die Bilder werden mit Hilfe der „Unicap“-Bibliothek (`http://unicap.sf.net/`) als Einzelbilder mit einer Bildwiederholrate von 35Hz von der Kamera geladen und im „Treiber“ verarbeitet:



(a) Unkalibriertes Bild (b) Kalibierungsvorgang (c) Kalibriertes Bild

Abb. 5: Vorgehensweise zur Kalibrierung

Zunächst wird die Kalibrierung angewendet: Das Videobild muss mit
dem Bild des Videobeamers in Übereinstimmung gebracht werden. Da es zu umständlich und unflexibel wäre, die durch die Linsen des Beamers und der Kamera, sowie des Spiegels entstandenen Verzerrungen physikalisch exakt heraus zu rechnen, wird auf eine Näherung mit Hilfe des „Morph"-Algorithmus [Ham] zurückgegriffen.

Im zweiten Schritt teilt eine Schwellenwertfunktion das Bild in „beleuchtete" und „unbeleuchtete" Bereiche ein. Daraufhin wird Kanten-Erkennungsalgorithmus (siehe [Mis]) angewendet, der die Umrisse der Berührflächen als Polygone zurückliefert. Um versehentlich aufgestützte Hände, Blendlichter und Kratzer auf der Scheibe auszufiltern, werden zu große oder zu kleine Flächen ignoriert. Von den übrigen Polygonen werden jeweils die Umkreise berechnet, als Koordinaten der Druckpunkte werden die Mittelpunkte der Umkreise verwendet. Zusätzlich wird der Radius des Umkreises als ungefähres Maß der Druckstärke verwendet.

Um Benutzerprogramm und Bildverarbeitung auf verschiedenen Computern mit verschiedenen Betriebssystemen verwenden zu können, werden die aus dem Bild extrahierten Informationen über die Netzwerkschnittstelle (per UDP) verschickt. So werden auf dem zu benutzenden Computer keine Ressourcen für die Bildverarbeitung verbraucht, da diese z.B. auf einen speziell angepassten Kleincomputer ausgelagert werden könnte.

Die Informationen werden am Zielcomputer empfangen, dekodiert und zu Punktlisten für jedes Einzelbild kombiniert. Die Daten können bei Bedarf direkt nach dem Empfang verändert und vorgefiltert, sowie um Zusatzinformationen erweitert werden, die durch den weiteren Programmverlauf erhalten bleiben.

Alternativ können die folgenden beiden Schritte übersprungen, und lediglich das entzerrte und mit der Schwellenwertfunktion bearbeitete Bild in der Benutzer-Anwendung empfangen werden.

Spuren zuordnen Die im Rahmen dieser Arbeit entwickelte Bibliothek sortiert die zur empfangenen Punktlisten in „Spuren" der Finger um: von den bekannten Spuren wird die erste und zweite Ableitung nach der Zeit berechnet, um mit Hilfe der erhaltenen Geschwindigkeits- und Beschleunigungsvektoren die Spur um einen Zeitschritt fortsetzen zu können. Die aktuellen Koordinaten werden nun mit den für die Spuren vorhergesagten verglichen – befindet sich ein Punkt nahe genug am wahrscheinlich nächsten Punkt einer Spur, so wird er ihr hinzugefügt. Spuren, denen über einen bestimmten Zeitraum keine neuen Punkte hinzugefügt wurden, werden gelöscht.

Dies ermöglicht bereits eine Vielzahl von Anwendungen, denn die Spuren liefern nicht nur die aktuellen Positionen der Finger, sondern auch die vergangenen sowie weitere, benutzerdefinierte Zusatzinformationen.

Finger gruppieren Mit Hilfe eines Segmentierungsalgorithmus können die Koordinaten in Gruppen zusammengefasst werden, was nützlich für die Gestenerkennung ist. Die Beschreibung des verwendeten Algorithmus würde allerdings den Rahmen dieser Arbeit sprengen, er kann am Wettbewerbsstand eingesehen werden.

3 Grundlegende neue Bedienkonzepte

3.1 Mehrere Mauszeiger

Die offensichtlichste Neuheit stellt gleichzeitig auch das größte Problem dar: es existieren noch keine „Mehrzeiger"-Anwendungen, vor allem deshalb, weil kein Oberflächen-Toolkit (etwa die Windows-Steuerelemente) für mehrere Mauszeiger ausgelegt ist. Auch fehlt in den Betriebsystemen selbst die Unterstützung, schließt man z.B. mehrere USB-Mäuse an einen Computer an, so werden deren Geschwindigkeitsinformationen einfach addiert. Für Linux steht mit dem „Multi-Pointer X-Server (`mpx`)" [Hut] allerdings ein Grafikserver bereit, der mehrere Mäuse unterstüzt. Davon profitieren kann man bereits dadurch, dass mehrere unterschiedliche Anwendungen gleichzeitig benutzt werden können. Meine Software unterstützt `mpx`, indem sie für jeden Finger einen eigenen PS/2-Touchscreen simuliert, dessen Koordinaten von `mpx` ausgelesen werden.

3.2 Lageunabhängigkeit

Traditionell sitzen Benutzer immer an der gleichen Position vor einem Bildschirm, sämtliche Software ist darauf ausgerichtet, dass „rechts oben" in Bildschirmkoordinaten auch „rechts oben" für den Betrachter darstellt. Anders wäre auch die Bedienung mit der Maus kaum möglich, sie kann nicht einfach z.B. um 90° gedreht verwendet werden, da sie dann auch entsprechend gedreht bedient werden müsste.

Ein horizontal positionierter Multi-Touchscreen ist jedoch, wie z.B. die Beispiele im nächsten Kapitel zeigen, komplett unabhängig von der Position des Benutzers.

3.3 Mehrere Benutzer

Aus den beiden obigen Konzepten folgt, dass beliebig viele Benutzer gleichzeitig – einen ausreichend großen Touchscreen vorausgesetzt – gemeinsam die gleiche Anwendung bedienen könnten. Bisher ist dies nur über die relativ wenig verbreitete Technik des „Remote Desktop" möglich oder benötigt spezielle Anwendungsprogramme, etwa ein „Whiteboard"-Zeichenprogramm oder den Texteditor „SubEthaEdit" (`http://www.codingmonkeys.de/subethaedit/`). Allerdings konnte gemeinsame Arbeit bisher immer nur an verschiedenen Computern über ein Netzwerk stattfinden oder mehreren Benutzer mussten sich eine Maus an einem Arbeitsplatz teilen.

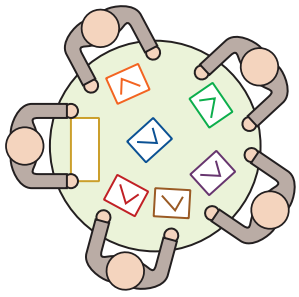


Abb. 6: Mehrere Benutzer an einem runden Mehrfinger-Touchscreen

3.4 Gleichzeitigkeit

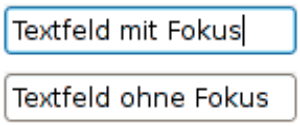


Abb. 7: Zwei Eingabefelder, eines fokussiert

Sämtliche grafischen Bedienoberflächen verwenden das Prinzip des „Fokus“. Dies bedeutet, dass sämtliche Bedienelemente zunächst aktiviert werden müssen, um sie Änderungen an ihnen vorzunehmen. Im Beispiel kann etwa in das obere Feld Text eingegeben werden, weniger offensichtlich ist z.B., dass „fokussierte“ Schieberegler auch mit dem Mausrad bedient werden können.

Wie etwa das Bézierkurven-Beispiel zeigt, wird dieses Prinzip durchbrochen, indem alle Elemente gleichzeitig bedient werden können.

Dies führt allerdings zu einem Problem bei Tastatureingaben: denn alle herkömmlichen Oberflächen senden Tastendrücke nur an das zur Zeit „fokussierte“ Element, ansonsten würden Eingaben in einem Fenster mit zwei Textfeldern nicht im gewünschten Feld, sondern in allen ankommen. Mögliche Lösungen wären z.B. die von Windows für „Tablet PC“ bekannte Handschrifteingabe zu verwenden oder das bekannte Fokusmodell für bestimmte Bedienelemente beizubehalten. Gleichzeitige Eingaben erforderten dann mehrere (reale oder virtuelle) Tastaturen.

4 Beispielanwendungen

In diesem Abschnitt werden einige Beispielanwendungen vorgestellt, die neuartige Bedienkonzepte in Mehrfinger-Touchsystemen demonstrieren sollen. Sie wurden alle von mir in C++ geschrieben und verwenden OpenGL (`http://www.opengl.org/`) für die graphische Darstellung.

Die statischen Bilder geben naturgemäß nicht den kompletten Eindruck der interaktiven Anwendungen wieder, auch wenn bei den folgenden Beispielen oft Bilderserien zur Veranschaulichung verwendet werden. Allerdings werden am Wettbewerbsstand sämtliche Programme demonstriert und können natürlich auch selbst ausprobiert werden.

4.1 Wiederkehrende Elemente der Programmsteuerung

Um die Software einheitlicher zu gestalten, finden sich einige Elemente in mehreren Programmen wieder:

4.1.1 Drehen, Skalieren und Verschieben

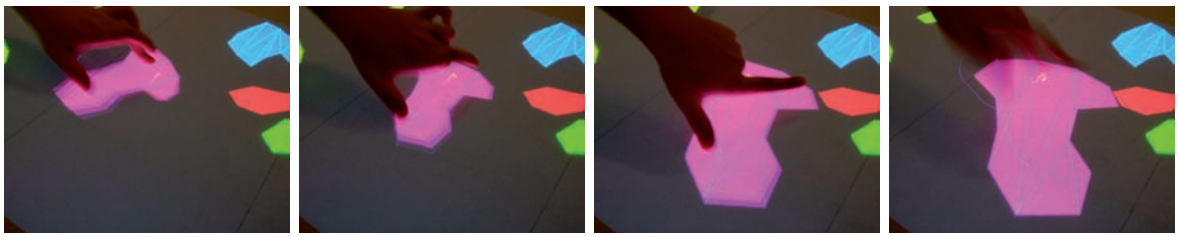


Abb. 8: Ein Objekt wird transformiert

Dank der Mehrfinger-Technik können Programme intuitiver bedient werden: so existiert keine Trennung zwischen den in Bildbearbeitungsprogrammen üblichen Werkzeugen „Drehen“, „Skalieren“ und “Verschieben“ mehr, diese Operationen sind mit einer einzigen Handgeste kombiniert möglich. Die mathematischen Grundlagen können am Wettbewerbsstand eingesehen werden.

4.1.2 Auswahl der Werkzeuge

Auf Grund des Mehrbenutzer-Charakters der Software wäre es kontraproduktiv, die Auswahl der virtuellen Werkzeuge für alle Benutzer gleichermaßen gelten zu lassen, da nicht alle Anwender das gleiche Werkzeug benutzen wollen. Daher wurde auf die in Bildbearbeitungssoftware übliche „Toolbox“ gänzlich verzichtet – die Werkzeugwahl geschieht durch verschiedene „Fingerkonstellationen“.

So steht etwa im Garten-Planungsprogramm ein einziger Finger für den „Pinsel“, zwei Finger nebeneinander gezogen jedoch für den “Radierer“.

4.1.3 „Pie“-Menüs

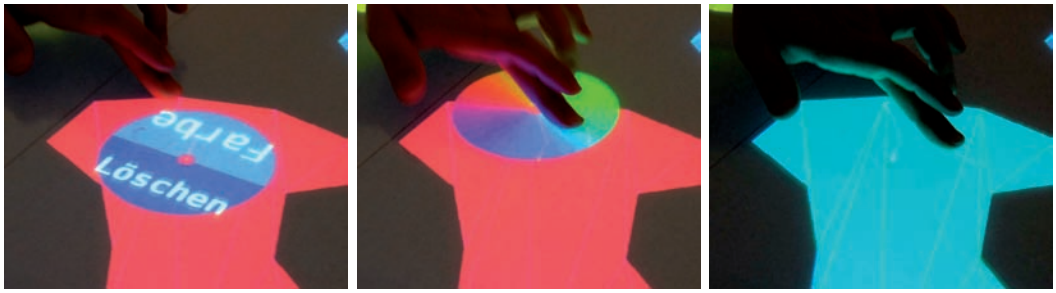


Abb. 9: Bedienung eines Pie-Menüs

Um dem Grundsatz der Lageunabhängigkeit zu entsprechen, werden statt der bekannten als vertikale Listen dargestellten Kontextmenüs, runde, so genannte „Pie“-Menüs verwendet. Dank der kreisförmigen Anordnung der Menüpunkte benötigen sie wenig Bildschirmfläche und die Auswahl kann mit einfachen Gesten direkt am Objekt getroffen werden.

4.2 Anwendungsprogramme

4.2.1 Mind-Mapping

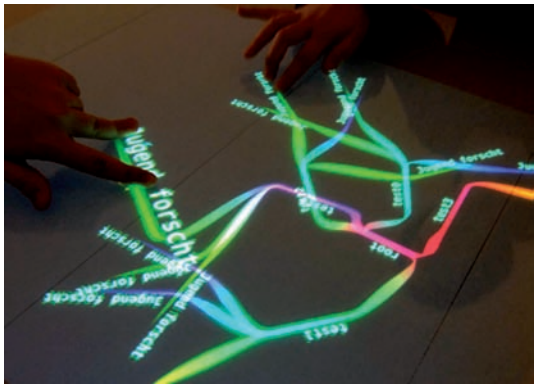


Abb. 10: Foto des Mind-Mapping-Programms

In vielen Bereichen erfreuen sich so genannte „Mind Maps“ großer Beliebtheit: oft werden diese „Gedankenkarten“ gemeinsam während einer Besprechung an einer Tafel gezeichnet. Herkömmliche Programme, die dieses System auf den Computer übertragen wollen, unterliegen den bekannten Einschränkungen: sie sind nur umständlich von einem einzigen Benutzer bedienbar.

Daher wurde im Rahmen dieser Arbeit eine Software entwickelt, die es, etwa wenn der Mehrfinger-Touchscreen als Konferenztisch eingesetzt wird, mehreren Benutzern gleichzeitig ermöglicht, die Mind Map zu bearbeiten.

4.2.2 Malprogramm

Dieses Programm erlaubt einfache Formen zu erstellen, indem mit dem Finger eine geschlossene Spur gezogen wird. Die umschlossene Fläche wird dann mit der Farbe der Spur gefüllt und im Programm als eigenständiges Objekt gespeichert. Dieses kann dann wie in 4.1.1 beschrieben bearbeitet werden. Um die Objekte weiter verändern zu können, kann auf die oben vorgestellten Menüs zurückgegriffen werden.

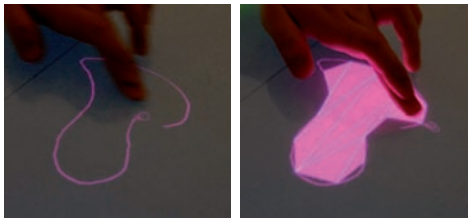
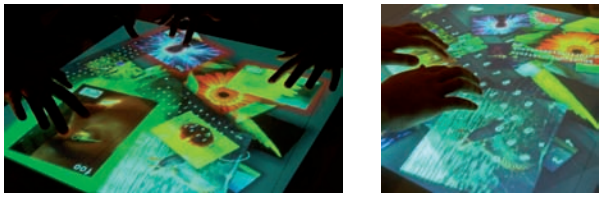


Abb. 11: Eine Form wird gezeichnet

4.2.3 Bildverwaltung



(a) Das Programm wird von zwei Benutzern gleichzeitig bedient

(b) Eingeblendete Tastatur

Abb. 12: Fotos der Bildverwaltungssoftware

Dieses einfache Bildverwaltungsprogramm verwendet die gleiche Technik wie das Malprogramm um Bilder verschieben, drehen und skalieren zu können. Während auch herkömmliche Software ermöglicht, Bilder zu kategorisieren indem sie auf Stapel geschoben werden, können hier – einen genügend großen Touchscreen vorausgesetzt – beliebig viele Benutzer gleichzeitig gemeinsam Bilder manipulieren und betrachten.

Jedem Bild ist eine Bildschirmtastatur zugeordnet, die bei Bedarf per Menü eingeblendet werden kann. Dies löst das Problem des nicht mehr vorhandenen Tastaturfokus und ermöglicht, mehrere Objekte gleichzeitig zu beschriften.

Abb. 13: Vergrößern, Drehen und Verschieben eines Bildes mit einer Handbewegung

4.2.4 Gartenplaner

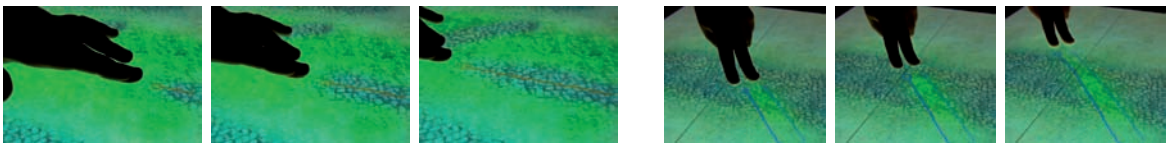


Abb. 14: Fotos des Garten-Planungsprogrammes

Ein möglicher Einsatzort dieses Programmes wäre ein Architekturbüro, in dem etwa eine Familie gemeinsam und flexibel ein Grundstück planen kann. Mit Hilfe der in 4.1.2 beschriebenen Techniken, können z.B. Grasflächen und Wege angelegt werden, per „Drag and Drop“ (Ziehen und Fallen lassen) werden Pflanzen positioniert.

4.2.5 Bézierkurven

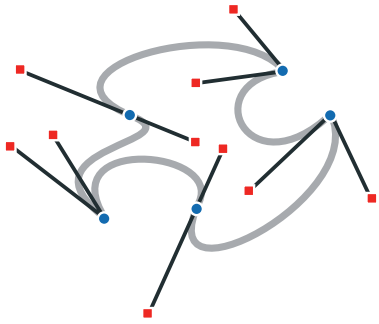


Abb. 15: Eine Bézierkurve (grau) mit Eckpunkten (blau) und Steuerpunkten (rot)

Bézierkurven sind besonders in der Vektorgrafik auf Grund ihrer großen Flexibilität beliebt und weit verbreitet: Im Gegensatz zu Polygonen sind jedem Punkt einer Bézierkurve zwei Steuerpunkte zugeordnet, mit denen der Verlauf der Kanten beeinflusst werden kann. Auf diese Weise können mit wenigen Punkten komplexe, Formen beschrieben werden.

Um mit herkömmlicher Grafiksoftware eine Bézierkurve (auch „B-Spline“ genannt) zu bearbeiten wird immer gleich vorgegangen: es wird ein einzelner Eckpunkt durch Klicken ausgewählt und verschoben. Für den jeweils gewählten Punkt werden die vier Steuerpunkte der beiden Kanten angezeigt, diese können wieder mit der Maus verschoben werden. Größere Änderungen sind so nur mit vielen Mausklicks zu verwirklichen.

Das kleine Beispielprogramm soll die Vorteile beliebig vieler Zeiger demonstrieren: Da beliebig viele Eck- und Steuerpunkte gleichzeitig verschoben werden können, ist theoretisch jede Änderung in einem einzigen Arbeitsschritt durchführbar. Dies erleichtert auch die Feinanpassung durch „zupfen“ an der Kurve und führt daher bei geringerem Zeitaufwand zu besseren Ergebnissen.

Zur Feinselektion kommen die im Rahmen ihres Diplomprojektes „tangent“ von Christian Iten und Daniel Lüthi erfundenen „Digital Tweezers“ (digitale Pinzette, [IL07]) zum Einsatz: Die „Pinzette“ kann mit zwei Fingern bewegt und mit einem dritten aktiviert werden, wobei nicht die realen Positionen der beteiligten Finger erfasst werden sondern lediglich die Spitze des Hilfsmittels. Da der zu selektierende Bereich so nicht von den Fingern des Benutzers überdeckt wird, ist eine deutlich präzisere Auswahl nahe nebeneinanderliegender Punkte möglich, die sonst nur durch langsame Mausbewegungen erreicht werden könnte.

4.2.6 Strömungssimulationen

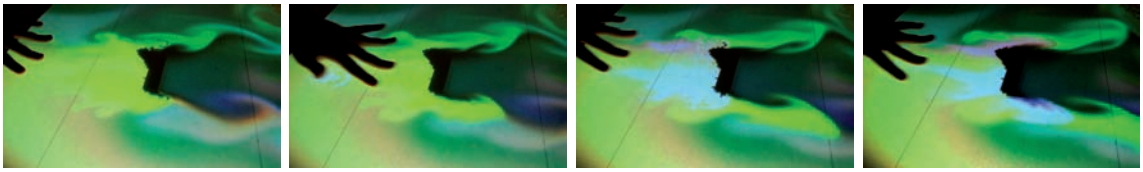


Abb. 16: Ein Hindernis wird umflossen

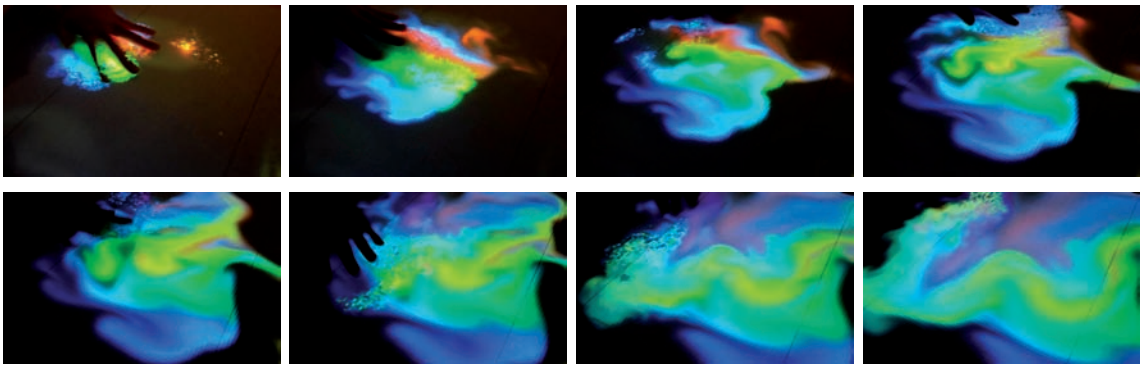
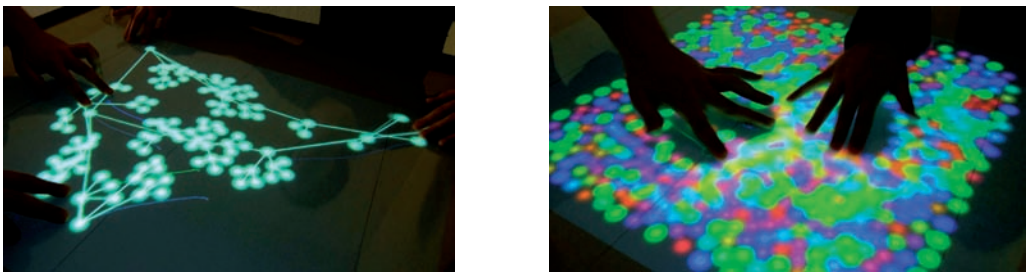


Abb. 17: Virtuelle Farbe wird umgerührt

Vor allem im Fahrzeugbau ersetzen Strömungssimulationen zunehmend Tests im Windkanal. Während „professionelle“ Simulationen auf Supercomputern berechnet werden, können sie niedriger aufgelöst auch auf modernen Grafikkarten interaktiv laufen. Dieses Programm soll eine derartige interaktive Simulation demonstrieren: Die Benutzer können durch Berührungen wahlweise Farbe in die simulierte Flüssigkeit mischen oder Hindernisse in den Strom bringen. Ähnlich wie im Malprogramm können Formen gezeichnet und in den Flüssigkeitsstrom gebracht werden. Alternativ lassen sich, indem das Videobild direkt ausgewertet wird mit den Händen Hindernisse formen, etwa um die aerodynamischen Eigenschaften verschiedener Körper vorzuführen.

4.2.7 Graph-Layout



(a) Zwei Bäume werden entwirrt (b) Die Fingern stoßen die Partikel ab

Abb. 18: Zwei Anwendungsmöglichkeiten des Layout-Programmes

In der Informatik ist ein „Graph“ ein aus Knotenpunkten („•“) und Kanten („ ⟶ “) bestehendes Gebilde, in dem beliebige Knoten durch beliebig viele Kanten miteinander verbunden werden können, z.B. „ •⇄•⇄• “.

Da die Koordinaten der Knotenpunkte oft nicht vorgegeben sind, wird versucht, den Graphen automatisch möglichst sinnvoll darzustellen. Dazu kann der iterative „Spring Embedder“-Algorithmus verwendet werden. Auf einem einfachen physikalischen Modell basierend versucht er, die Abstände zwischen den Knoten zu vergrößern, gleichzeitig aber die Längen der Kanten zu verkleinern. So würde etwa der obige Graph nicht in einer Zeile sondern eher als Dreieck dargestellt.

In meinem Programm kann der Benutzer in den Anordnungsprozess eingreifen, indem einzelne Knoten mit den Fingern „festgehalten“ und verschoben werden können. So lassen sich z.B. bestimmte Anordnungsprobleme, die der Algorithmus nicht lösen kann, manuell korrigieren. Ausserdem lassen sich physikalisch ausreichend korrekte Seile und Stoffe simulieren, mit denen die Benutzer fast wie mit ihren realen Äquivalenten interagieren können (z.B. zwei Hälften mit beiden Händen auseinanderziehen um den Stoff zu spannen). Dies wäre mit einer Maus nicht ohne weiteres möglich.

4.2.8 Skulpturen

Abb. 19: Fotos des Skulpturprogramms

Dieses Programm ermöglicht den Benutzern, dreidimensionale Skulpturen zu erstellen. Dabei können mit den Fingern Linien gezeichnet werden, die allerdings nicht in der Zeichenebene verbleiben sondern sich um die Hochachse des Objektes drehen. So kann etwa ein Kreis mit der Achse als Mittelpunkt durch bewegungsloses Berühren des Bildschirmes an einem Punkt gezeichnet werden, eine spulenartige Form entsteht, indem der Finger stetig in kleinen Kreisen bewegt wird.

4.3 Spiele

Die Computerspieleindustrie hat sich zu einem milliardenschweren Wirtschaftsfaktor entwickelt und stellt oftmals die treibende Kraft für die Hardwareentwicklung dar. So sind es z.B. moderne Computerspiele, die immer leistungsfähigere Grafikhardware verlangen.

Vor diesem Hintergrund werden im folgenden einige neuartige Spielkonzepte, die mit der bisherigen Technologie nicht möglich waren, vorgestellt.

4.3.1 Khronos Projector

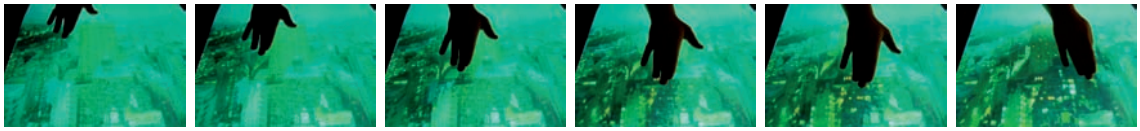


Abb. 20: Bildschirmfotos des „Khronos Projector“ (mit dem übernommenen „Jefo“-Video)

An der University of Tokio wurde ein „Khronos Projector“ [CI05] genanntes System entwickelt, mit dem Videos nicht mehr nur „linear“ angesehen werden können. Die Idee ist, dass es sich bei einem Video um ein 3D-Bild in Quaderform handelt, aus dem nur eine Ebene die sich kontinuierlich auf der Zeitachse weiterbewegt „ausgeschnitten“ und angezeigt wird. Der „Khronos Projector“ erlaubt nun, nicht mehr nur zur „Vorderseite“ parallele Ebenen (also „gleichzeitige“ Pixel) aus dem Video anzuzeigen, sondern beliebige Freiflächen im Raum. Die Benutzer können durch Berühren des Bildschirmes diese Schnittfläche verändern, dem aufgewandten Druck entsprechend wölbt sie sich nach „hinten“ („später“) und fließt langsam wieder zurück.

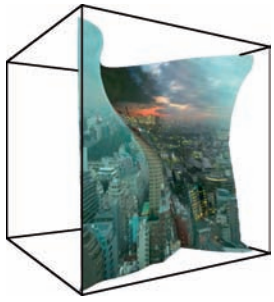


Abb. 21: Der Videowürfel, die Zeit nimmt nach hinten zu

Dieses System wurde, da die Originalquellen leider nicht zur Verfügung standen, nachprogrammiert, wobei die Möglichkeiten moderner Grafikkarten ausgeschöpft werden, indem der Großteil der Berechnungen auf dem Grafikprozessor stattfindet.

4.3.2 Boids

Der 1986 von Craig Reynolds entwickelte „Boids“-Algorithmus (siehe [Rey]) zur Schwarmsimulation erfreut sich auch heute noch einiger Beliebtheit, vor allem wegen seines einfachen Aufbaus: Die Schwarmwesen – hier als Dreiecke dargestellt – bewegen sich mit einer bestimmten Geschwindigkeit in eine bestimmte Richtung fort. Dabei passen sie Geschwindigkeit und Richtung nach einem einfachen Regelwerk ständig an ihre Umgebung (die Wesen in einem bestimmten Umkreis) an: 1. Nachbarn nicht anrempeln, 2. In die Durchschnittsrichtung der Nachbarn fliegen, 3. In die Mitte der Nachbarn fliegen.

Meine (zweidimensionale) Implementierung von „Boids“ simuliert einen Schwarm mit 1000 Wesen nach den genannten Regeln und registriert zusätzlich die Finger der Benutzer als „Superwesen“, an denen sich die anderen mit hoher Wahrscheinlichkeit (abhängig vom Druck) orientieren. So ist es möglich, den Schwarm zu „steuern“, indem man ihm eine Hauptrichtung vorgibt oder mehrere Finger in einem großen Kreis bewegt.

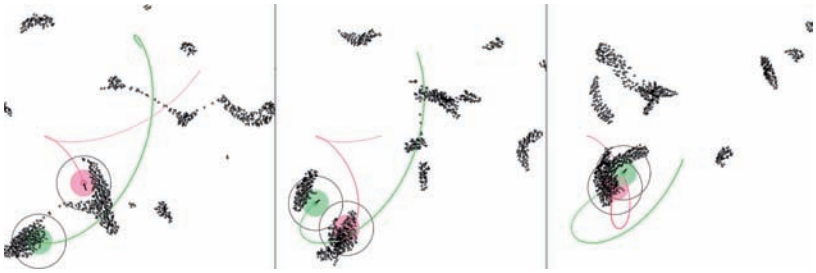


Abb. 22: Bildschirmfotos des „Boids“-Simulationsprogrammes, im Abstand von je einer Sekunde

4.3.3 Pong

Abb. 23: Pong mit zwei Spielern

Auch das erste Computerspiel der Welt – „pong“ – wird immer noch „weiterentwickelt“. In meiner Mehrfinger-Version des Spiels besteht keine Einschränkung auf eine bestimmte Bildschirmhälfte oder gar die Anzahl der Spieler: werden zwei Finger ungefähr gleichzeitig in geringem Abstand aufgedrückt, so entsteht zwischen ihnen ein neuer „Schläger“. Die Schläger folgen den Bewegungen der Finger, behalten allerdings aus Fairnessgründen ihre Länge bei.

4.3.4 „Marble Market“

Abb. 24: Bilderserie des Spielverlaufs

Dieses Spiel wurde von den bereits erwähnten „SmartSkin“-Entwicklern erfunden. Es werden Murmeln auf einem unebenen Feld, das die Spieler durch Druck auf den Touchscreen manipulieren können (wo gedrückt wird, hebt sich das Feld an), simuliert. Je nach Spielmodus müssen die Spieler das ihnen zugeordnete Tor vor den Kugeln verteidigen oder versuchen möglichst viele Murmeln in das eigene Tor zu leiten. Dabei können die Spieler ausschließlich das Höhenfeld verändern, es existieren keine Spielfiguren und die Murmeln folgen nur dem Verlauf des Feldes.

Auch hier wird das Videobild „direkt“ ausgewertet, da die Verfolgung der Spuren unnötig wäre.

5 Ergänzungen und Ausblicke

Mittlerweile umfasst der im Rahmen dieser Arbeit geschriebene Programmcode etwa 50.000 Zeilen und erreicht damit die Obergrenze des für einen einzelnen Entwickler Überschaubaren. Daher wurde sämtliche für diese Arbeit entwickelte Software unter der „GNU General Public License“ freigegeben, auch in der Hoffnung, die Entwicklung einer Standardbibliothek für Mehrfinger-Software, und damit die Etablierung von Mehrfinger-Touchscreens, zu beschleunigen. Sie ist unter `http://multitouch.sf.net/` abrufbar und wird auch nach Abschluss dieser Arbeit noch weiter entwickelt werden.

Aller Voraussicht nach werden sich Mehrfinger-Touchscreens erst dann auf dem Markt durchsetzen, wenn große Softwarehersteller Programme zur Verfügung stellen, die diese Technik nutzbar machen. Allerdings ist es unwahrscheinlich, dass die zukünftigen Touchscreens auf derselben Technik wie der hier vorgestellte basieren werden, da diese zwar die günstigste in Eigenregie herstellbare ist, in der Praxis werden sich aber wohl andere Technologien mit geringererem Platzbedarf etablieren.

Danksagung

Ich bedanke mich herzlich bei

meiner Familie die erhebliche Einschnitte in ihre gewohnte Lebensführung geduldig ertrugen und mich tatkräftig unterstützten

Claus-Peter Horn, meinen Betreuungslehrer, für langjährige gute Unterstützung

The Imaging Source Europe als Hauptsponsor meiner Arbeit, der die Infrarotkamera samt Zubehör kostenlos zur Verfügung stellte (`http://www.theimagingsource.biz/`)

Reichmann Feinoptik für den Tageslicht-Sperrfilter (`http://www.reichmann-feinoptik.de/`)

Literatur

[CI05] Cassinelli, Alvaro ; Ishikawa, Masatoshi: Khronos Projector. In: *Emerging Technologies, SIGGRAPH 2005*. Los Angeles : ACM Press, 2005

[DL01] Dietz, Paul ; Leigh, Daren: Diamondtouch: A multi-user touch technology. In: *Proceedings of UIST 2001*, ACM Press, 2001

[Ham] Hammerhead Studios: *Beschreibung des Morph-Algorithmus (Siggraph '92)*. `http://www.hammerhead.com/thad/morph.html`

[Han] Han, Jefferson Y.: *Multi-Touch Interaction Research*. `http://cs.nyu.edu/~jhan/ftirtouch/`

[Han05] Han, Jefferson Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM Press, 2005. – ISBN 1–59593–271–2, S. 115–118. – Dieses Paper stand mir nicht zur Verfügung, da es eine teure ACM-Mitgleidschaft erfordern würde.

[Hut] Hutterer, Peter: *Multi-Pointer X-Server*. `http://wearables.unisa.edu.au/mpx/`

[IL07] Iten, Christian ; Lüthi, Daniel: The Digital Tweezers – A Precise Selection Technique for Multi-Touch Screens. In: *TEI 2007 – First International Conference on Tangible and Embedded Interaction*, 2007

[Mis] Miszalok, Volkmar: *Beschreibung des Crack-Codes*. `http://miszalok.de/Lectures/L08_ComputerVision/Computer_Vision_deutsch.htm`

[Rek02] Rekimoto, Jun: Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, 2002, S. 113–120

[Rey] Reynolds, Craig: *Boids Flocking Algorithm*. `http://www.red3d.com/cwr/boids/`

[Wil04] Wilson, Andy: Touch Light: An Imaging Touch Screen and Display for Gesture-Based Interaction. In: *International Conference on Multimodal Interfaces*, Microsoft Research, 2004