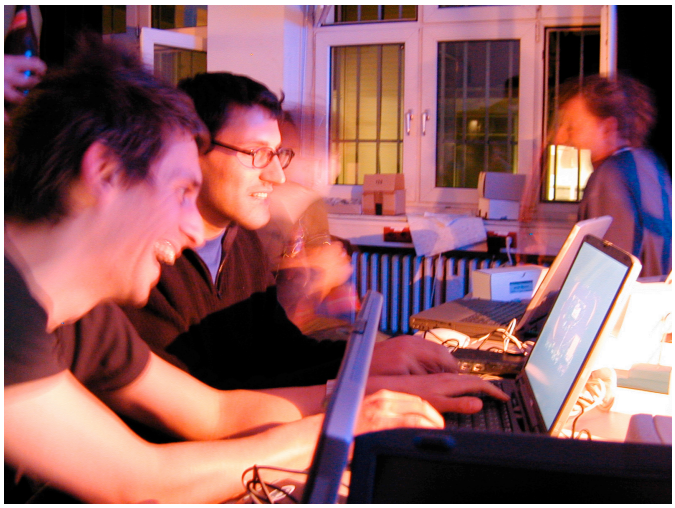
CODEKIT – THEORIE UND PRAXIS

Pionier C. Babbage stanzte die erste Lochkarte von Hand.

INHALT:

ÜBERSICHT



CodeKit im Einsatz: Javakurs an der KHM

KIT – EIN MODELLVERSUCH

Das an der Kunsthochschule für Medien in der Fächergruppe Kunst- und Medienwissenschaft angesiedelte Modellprojekt KIT untersucht die Wechselwirkung von künstlerischer Praxis und neuen Technologien unter der Perspektive einer künstlerischen Ausweitung und Modifizierung digitaler Werkzeuge. Das Projekt verbindet theoretische und praktische Fragestellungen, die im Lehrangebot der Kölner Kunsthochschule für Medien aufgegriffen und in theoretischen Grundlagentexten erörtert werden. KIT ist Teil des bundesweiten Modellversuchsprojekts KuBiM [http://www.kubim.de] (Kunst und Bildung im Medienzeitalter) und besteht aus den Modulen TextKit [http://www.khm.de/kmw/kit] und CodeKit [http://java.khm.de].

CODEKIT - ENTWICKLUNG EINES JAVA-PROGRAMMIERKURSES FÜR KÜNSTLERISCHE AUSBILDUNGSKONTEXTE

Das Modul CodeKit geht der Frage nach, welche Bedeutung die Programmierung für die künstlerische Praxis hat und stellt ein didaktisches Konzept dar, um Künstler und Künstlerinnen an die Programmierung heranzuführen. Das CodeKit ist ein frei zugängliches Java-Programmarchiv und dient einerseits als Grundlage für die Programmierkurse an der Kunsthochschule für Medien und andererseits als Materialsammlung und Experimentierfeld in der künstlerischen Projektproduktion.

CODEKIT@NETZSPANNUNG.ORG - SINN UND ZWECK

Die Präsentation des Modellversuchs auf Netzspannung.org will anhand von Texten, Projektdokumentationen und einem „Tutorial“ die Theorie und Praxis, wie sie kennzeichnend für die Arbeit mit dem CodeKit ist, vorstellen und veranschaulichen. An den für die Präsentation ausgewählten KHM Projekten „Framedrucker“, „Update“, „Bitmirror“ und „Loser Raum“ wird besonders gut

deutlich, in welchen unterschiedlichen Formen Programmierung im künstlerischen Kontext eingesetzt werden kann. Strategien und Arbeitsprozesse werden sichtbar, die Verknüpfung von angeleitetem Lernen und eigenständiger künstlerischer Arbeit aufgezeigt.
Ferner führt das Tutorial als „Selbstlernmodul“ in das Arbeitsumfeld des CodeKits ein. Schrittweise wird ein neues Experimentierfeld eröffnet, das eigenständiges kreatives Arbeiten und weiterführende Projektarbeit im Medium Computer ermöglicht.

PERSONEN

Prof. Georg Trogemann

Dr. Jochen Viehoff

KONTAKT

Dr. Jochen Viehoff
Peter-Welter-Platz 2
viehoff@khm.de

Kunsthochschule für Medien Köln
50676 Köln

LINKS

Kunsthochschule für Medien Köln [http://www.khm.de]

CodeKit [http://java.khm.de]

Kit [http://www.khm.de/kmw/kit/]

Kubim [http://www.kubim.de]

labIII – Interface Labor [http://interface.khm.de]

EINLEITUNG

```
int width =640;
int height=480;
int dim = width*height;
```

Ausschnitt Java-Programmcode: Variablendeklaration

> „Technologie ist nichts anderes als die gegenständlich gewordene Widerspiegelung der menschlichen Seele in die Natur. Maschinen sind vom Menschen produziert. Sie sind nichts anderes als die Materialisierung dessen, was im Kopf, in der Psyche des Menschen bereits vorhanden ist. Maschinen können als materialisierte Projektionen von Wesensmerkmalen des Menschen begriffen werden. Nicht die Technik wäre dann das größte Problem des gegenwärtigen Menschen, sondern der Mensch selbst."

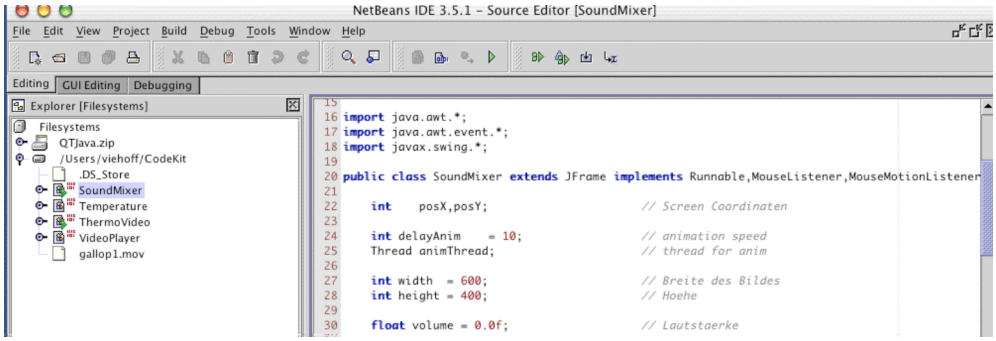
Arno Bammé (1983), in: Maschinen-Menschen Mensch-Maschinen: Grundrisse einer sozialen Beziehung, Rowohlt Taschenbuch, Hamburg, p. 110.

AUSGANGSPUNKT

Ausgangspunkt des im Folgenden vorgestellten Programmiermoduls CodeKit ist die Erkenntnis, dass unsere gegenwärtige Kultur in besonderer Weise durch Medien und die von ihnen geschaffenen Realitäten geprägt ist. Die neuen Informations- und Kommunikations-Technologien werden – wie jede Technologie - vor dem Hintergrund eines bestimmten, unausgesprochenen Verständnisses der Natur des Menschen entwickelt und verändern gleichzeitig unsere Verhaltensweisen. Entscheidend ist nun, dass weder die medialen Möglichkeiten, Formen und Konsequenzen dieser Technologien sich selbst offenbaren, noch geht die theoretische und ästhetisch-praktische Reflexion der Medienentwicklung quasi automatisch von statten. Es bedarf besonderer Anstrengungen und in der Bildung besonderer Situationen und Umfelder, sowie geeigneter Lehr- und Lernmittel, um den kritischen wie experimentellen Umgang mit neuen wie alten Medien zu ermöglichen. Das an der Kunsthochschule für Medien in der Fächergruppe Kunst- und Medienwissenschaft angesiedelte **Modellvorhaben KIT - Kunst – Informatik – Theorie**, das im Rahmen des BLK-Programms „Kulturelle Bildung im Medienzeitalter" (KuBiM) gefördert wurde, setzt an diesem Punkt an. Es untersucht die Wechselwirkung von künstlerischer Praxis und neuen Technologien unter der Perspektive einer künstlerischen Ausweitung und einer Modifizierung des Gebrauchs digitaler Werkzeuge. Insbesondere widmet es sich den noch nicht angemessen gewürdigten

engen Beziehungen zwischen Kunsttheorie und Informatik im Rahmen einer Entwicklung zeitgemäßer künstlerischer Praktiken mit und durch Medien.

PROGRAMMIERUNG



NetBeans: eine Java-Entwicklungsumgebung für das CodeKit.

Bei der Frage, welche prinzipiellen Kenntnisse und Fertigkeiten nötig sind, um sich mit den neuen computerbasierten Medien sowohl medienwissenschaftlich als auch künstlerisch-praktisch fundiert auseinandersetzen zu können, kommt nach unserer Überzeugung der Programmierung eine zentrale Stellung zu. Einige Gründe, warum der Programmierung eine Schlüsselstellung in dieser medialen Praxis eingeräumt wird, seien im Folgenden beispielhaft genannt.

VERSTEHEN

Kulturelle Bildung hat die Aufgabe, durch einen experimentellen Gebrauch der Medien zu einer realistischeren Einschätzung der Möglichkeiten und Grenzen neuer Medien zu kommen. Einen wirklich grundlegenden Zugang zu den neuen Informations- und Kommunikationstechnologien bietet aber nur die Ebene der Programmierung. Die Programmierung und die damit verbundenen algorithmischen Grundstrukturen können als Erklärungsprinzip für viele natürliche Erscheinungen in einem mechanistischen Weltbild herangezogen werden.

PRAXIS

Viele künstlerische Projekte – insbesondere avancierte medienkünstlerische Arbeiten - erfordern für ihre Realisierung Kenntnisse in der Programmierung. Ein wesentliches Ausbildungsziel muss deshalb darin bestehen, Studenten in die Lage zu versetzen, selbstständig an ihren Projekten zu arbeiten. Auch wenn natürlich nicht alle Medienkünstler Programmierer werden sollen, müssen sie dennoch so viel von den zugrunde liegenden Strukturen verstehen, dass sie den Aufwand und die Grenzen der Systeme mit denen sie umgehen, einschätzen können. Kenntnisse von den Grundstrukturen der Programmierung sind hierfür unabdingbar.

IMAGINATION – SPRACHE – MATERIAL

Die Herstellung medialer Inhalte und Anwendungen ist im Kern ein Prozess der Umformung. Das Imaginäre (Vorstellungen, Phantasien, Träume, Utopien) wird in Symbolisches, d.h. sprachliche Modelle transformiert. Dabei ist eine

Einlassung ins Reale unumgänglich. Material, das an sich ohne Bedeutung ist, wird semantisch aufgeladen. Die Relationen und Spannungen zwischen Imagination, Sprache und Material (in der Sprache des Computers: das Interface) wirken in der Programmierung in besonderer Weise zusammen. Programme sind nicht nur textlich verfasste Imaginationen und Denkvorgänge, als Beschreibungen besitzen sie zusätzlich eine feste Bindung ans Reale, da sie in direkter Weise Handlungsanweisungen für die Maschine darstellen und sich in reales Material einschreiben. Dieses Dreiecksverhältnis zwischen Imagination, Sprache und Material – das gleichzeitig ein Grundmuster jedes kulturellen Prozesses ist – tritt auf der Ebene der Programmierung besonders deutlich in Erscheinung und kann hier experimentell untersucht werden.

SCHULE DER KOGNITION

Es wird oft betont, dass die neuen multimedialen Formen die Grenzen zwischen den traditionellen künstlerischen Fachdisziplinen aufheben, da sie die Wahrnehmung mit allen Sinnen einfordern. Die Ebene der Programmcodes zielt dagegen auf eine andere Form der Generalisierung ab. Sie abstrahiert von den konkreten Medienausprägungen wie Bild, Text oder Ton und wirkt auf der Ebene der Denkprozesse und der Reflexion. Programmierer im Bereich der neuen Medien müssen permanent Denken und Wahrnehmen miteinander verbinden und Übersetzungsleistungen ins Visuelle und Akustische vollbringen. Den traditionellen Schulen der Wahrnehmung mit allen Sinnen muss deshalb beim Umgang mit neuen Medien eine Schule der Kognition an die Seite gestellt werden. Die Einführung in die Programmierung ist ein Ansatz in diese Richtung.

AUSBILDUNG IN DER PRAXIS

CodeKit: ein Javakurs für Künstler

ÜBERSICHT

Vorauszuschicken ist, dass die im Rahmen von CodeKit bereitgestellten Programmbeispiele - wie das gesamte Material - ursprünglich nicht als Selbstlernkurs konzipiert wurden. Der Inhalt von CodeKit wird an der Kunsthochschule für Medien Köln im Rahmen einer zweiwöchigen Blockveranstaltung mit dem Titel „Einführung in die Java-Programmierung“ vermittelt. Die Präsenz und Hilfestellung der Lehrenden halten wir generell für sehr wichtig, um einen effektiven und reflektierten Umgang mit neuen Medien zu gewährleisten. Im Zusammenhang mit der Programmierung gilt dies aufgrund der Komplexität des Stoffes in besonderem Maße.

MOTIVATION

KHM Javakurs: Arbeiten in kleinen Gruppen

Die Motivation für die Entwicklung eigener Lernmaterialien einer Einführung in die Java-Programmierung basiert auf unserer Erfahrung, dass Softwareentwicklung in künstlerischen Arbeitszusammenhängen von anderen Erwartungen und Zielsetzungen getragen wird, als die professionelle Softwaretechnik der Informatik. Softwaretechnik (engl. „software engineering“) ist, im Gegensatz zur künstlerischen Programmierung, ein stark standardisierter Prozess und im Wesentlichen an kommerziellen Kriterien orientiert. Sie befasst

sich mit der Ingenieursmäßigen Konstruktion und der professionellen Entwicklung großer Softwaresysteme, d.h. es stehen komplexe, qualitativ hochwertige Produkte und deren zuverlässiges Funktionieren im Vordergrund. Als Qualitätsmerkmale gelten neben Preis und Bedienerfreundlichkeit, die Korrektheit, Effizienz, Zuverlässigkeit und Anpassungsfähigkeit des Systems.

PROGRAMMIEREN ALS KÜNSTLERISCHE PRAXIS

Ganz anders ist die Situation der Programmierung als künstlerische Praktik. Der Computer wird hier als Werkzeug der Imagination eingesetzt. Es geht in erster Linie darum, Ideen zu testen und zu verwerfen oder aber weiterzuentwickeln. In der Kunst geht es im Unterschied zur Technik damit nicht vorrangig um die Lösung spezifischer Probleme, sondern um die Annäherung an Problemstellungen. Programmierung ist hier weniger ein Werkzeug, um ein klar definiertes Problem abzuarbeiten, sondern ein Werkzeug, um Probleme zu erarbeiten und Fragen aufzuwerfen. Es existiert auch keine Lösung, die nach allgemein akzeptierten Kriterien zu bewerten wäre. Es gibt keinen festzulegenden Punkt, an dem das Projekt „fertig“ wäre. Vielmehr wird die Entwicklung nach Erreichen eines zufriedenstellenden Status’ gestoppt. Die direkte Auseinandersetzung mit dem Code als Material und die Widerstände, die sich erst im Vorgang des Programmierens zeigen und überwunden werden müssen, sind hierbei wichtig. Der Prozess der Softwareerstellung ist damit in künstlerischen Zusammenhängen sehr viel individuelller als die Methoden, die durch die Softwaretechnik der Informatik bereitgestellt werden. Des Weiteren sind künstlerische Arbeiten stark an Interface-Fragen und audio-visuellen Anwendungen orientiert. Einführende Programmbeispiele, die sich an Bild, Ton und Interaktion orientieren, sind deshalb von sehr viel größerer Bedeutung als etwa die Eleganz und Effizienz (im Hinblick auf Speicherplatz und Zeitkomplexität) von Algorithmen, wie sie für die Informatik wichtig ist.

AUSGANGSPOSITIONEN

Diesen unterschiedlichen Ausgangspositionen von Informatik und Kunst muss eine Einführung in die Programmierung, die in künstlerischen Arbeitszusammenhängen erfolgreich sein will, Rechnung tragen. Die Inhalte des Kurses lassen sich in die drei Bereiche Sound, Bild und Interaktion unterteilen. Die einzelnen Beispiele aus diesen drei Bereichen greifen Grundprobleme der medialen Praxis auf. Die Lösungsbeispiele sind damit gewissermaßen Musterlösungen, die im Laufe des Kurses immer wieder neue Verbindungen miteinander eingehen und damit neue und oft überraschende Anwendungen ermöglichen. Die Erwartungen an die Kursteilnehmer ist dabei nicht, sie zu versierten Programmierern auszubilden, sondern sie an das Feld der Programmierung heranzuführen und dessen Potential erfahrbar zu machen. Dazu bekommen die Studenten einfache Programmversionen an die Hand, in die sie ihre eigenen Ideen und Versuche einschrieben können. Damit soll vor allem erreicht werden, dass sich rasch kleine Erfolge einstellen und die Lernenden kreativ mit den Codes umgehen und so von der ersten Stunde an eigene kleine Experimente entwickeln. Um effektives und individuelles Arbeiten und Experimentieren zu ermöglichen, achten wir auf Kursstärken zwischen 10 und 15 Teilnehmern bei 2 bis 3 Lehrenden. Tatsächlich kann aber erst die längerfristige

Programmierpraxis im Anschluss an den Kurs in Einzelfällen dazu führen, dass Studenten in den Monaten nach dem Kurs die Fähigkeit erlangen, eigene Programmierprojekte autonom zu entwickeln. Entscheidend für diesen besonderen Lernerfolg ist die individuelle Betreuung dieser Studenten in der Zeit nach dem Kurs.

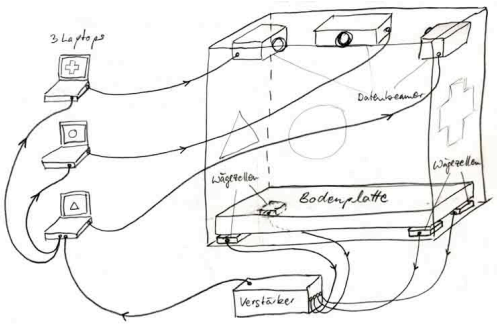
WERKZEUGKASTEN

Die Entwicklung des CodeKit verfolgt einen weiteren Nutzen, der über seine Verwendung im Programmierkurs hinausgeht. Es hat sich gezeigt, dass zwar nur wenige Kursteilnehmer den Sprung bis zur Entwicklung eigener kleiner Programmierprojekte schaffen, viele Studenten aber dennoch in der Folge künstlerische Arbeiten entwickeln, die Programmieranteile enthalten. CodeKit ist geeignet, die Realisierung dieser Projekte zu unterstützen. Wie in einem Werkzeugkasten lassen sich die Grundbausteine in ihrer Funktionalität kombinieren und so in neuen Kontexten wiederkehrende technische Probleme pragmatisch und effizient lösen. Die hier beschriebenen Projekte sind Beispiele für beide Verwendungsarten des CodeKit, einmal als Ausbildungskonzept und Rahmen für die eigenständige Entwicklung studentischer Projekte, zum zweiten als Werkzeugkasten, der es dem versierten Programmierer erlaubt, die im Rahmen medienkünstlerischer Projekte anfallenden Programmieraufgaben effizient und zielorientiert zu erledigen.

LOSER RAUM – ANJA KEMPE

Loser Raum: Am Rande der Balance

KURZBESCHREIBUNG

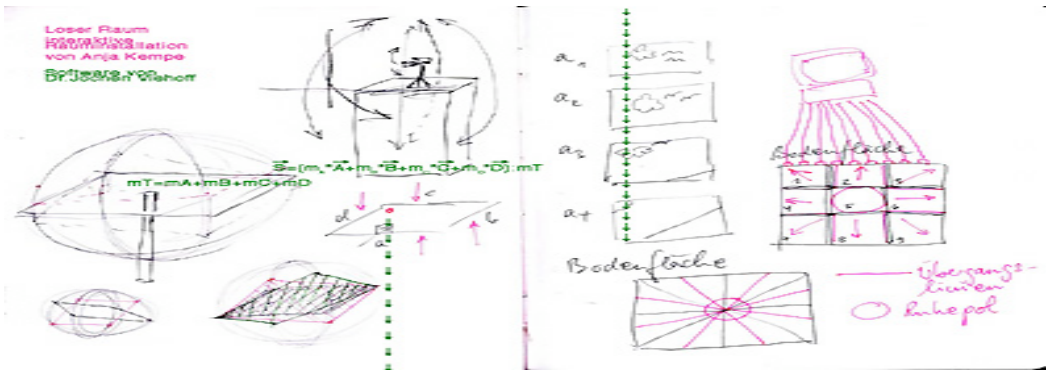


Der „Lose Raum“ ist eine interaktive Installation von Anja Kempe und entstand 2002/2003 im Interface-Labor der Kunsthochschule für Medien Köln. In der Installation werden die Fotografien von zwei bzw. drei Wänden eines Raumes auf die Wände selbst projiziert. Eine Wägeplattform misst die Gewichtsverteilung der Besucher in dem Raum und steuert durch Schwerpunktsverlagerungen das virtuelle Raummodell – die Projektion des Raumes deckt sich nicht mehr mit dem realen Raum, die virtuelle Welt kippt aus ihrer Gleichgewichtslage heraus.

„O-Ton“ der Künstlerin

„Loser Raum“ thematisiert den Körper in einer virtuellen Situation.
Es wird eine Fiktion vorgegeben: Ein schwankender Raum, der sich je nach dem Gewicht seiner Besucher hebt und senkt. Das Schwanken erfolgt über die Bewegung von drei Projektionen. Drei Projektionen bilden auf drei Wänden das Bild der Wände ab. Im Ruhezustand sind Wand und Bild der Wand deckungsgleich und man nimmt das projizierte Bild nicht wahr.
Die Fiktion wird nun zu einem körperlichen Erlebnis, weil die Projektionen an den drei Wänden auf die Bewegungen des Besuchers reagieren. Wie auf einem großen Floß kommt der Raum ins Schwanken, wenn auf dem Boden kein Gleichgewicht herrscht. Wenn allerdings alle Besucher gleichmäßig auf dem Boden verteilt sind, können Projektion und Wand wieder zur Deckungsgleichheit gebracht werden. Bewegen sich die Besucher, schwankt die Projektion entsprechend und erst wenn niemand mehr auf dem Boden herumläuft, kommt die bewegte Projektion langsam zur Ruhe. Der Raum atmet auf.

Entstehungsgeschichte



Loser Raum: Entwurfsskizze

Die interaktive Installation „Loser Raum“ ist ein konkretes Beispiel für die enge Zusammenarbeit zwischen Künstler, Programmierer und Techniker. Die Konzepte sowohl für das Interface-Design (Wägeplattform s.u.) als auch für die Simulation einer zweidimensionalen „Wippe“ entstanden unter Berücksichtigung physikalischer Kräfte. Weil sich das Projekt zudem nur durch eine aufwändige Programmierarbeit realisieren ließ, setzte die Künstlerin sich mit den Grundzügen

der Java-Programmierung auseinander und nahm am Programmierkurs und weiterführenden Fachseminaren teil. Diese Art der wechselseitigen Beeinflussung halten wir als „künstlerisches Experiment“ für sehr wichtig.

Anja Kempe hat mit der Installation „Loser Raum“ den Digital Sparks Preis 2002 in der Rubrik „Interaktive Kunst“ gewonnen. [http://netzspannung.org/digital-sparks/03/]

HARDWARE, FUNKTION UND AUFBAU

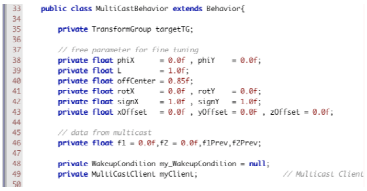
Steuereinheit für die Wägeplattform, Scherwägezelle

Für die Installation „Loser Raum“ müssen im Vorfeld drei Wände eines geeigneten Raumes abfotografiert werden. Ferner werden unter der Decke des Raumes drei Beamer so befestigt, dass mit Weitwinkelobjektiven jede Wand vollständig ausgeleuchtet wird. Jeweils ein Laptop generiert die Bildprojektion für einen Beamer. Das projizierte Bild des Raumes deckt sich mit dem realen Raum, solange kein Betrachter das Gleichgewicht stört.
Für die Interaktion ist in der Mitte des Raumes eine 4m x 4m große Wägeplattform als begehbarer Fußboden installiert (technische Daten s.u.). Vier präzise Wägezellen messen die Gewichtsverteilung von Personen auf der Plattform. Nach einfachen Regeln lässt sich daraus der Schwerpunkt der Gesamtmasse berechnen. Diese Gewichtskraft wirkt am Ort des Schwerpunkts auf die Simulation einer physikalischen zweidimensionalen Wippe, deren Aufhängungspunkt in der Mitte des Raumes liegt. Bedingt durch die äußere Kraft, neigt sich die Wippe. Die Auslenkung aus der Ruhelage wird von zwei unabhängigen Winkeln beschrieben. Korrespondierend zu der Auslenkung werden die Wandprojektionen aus ihrer Nullposition ausgeschwenkt – der gesamte virtuelle Raum ist in Bewegung. Lineare Rückstellkräfte sorgen dafür, dass die Wippensimulation (und der fest damit verknüpfte virtuelle Raum) langsam in ihre Ausgangposition zurück schwingt (Relaxation). Trägheit der Wippe und die Viskosität ihrer Umgebung legen die Dynamik der Bewegung anhand weniger Parameter fest.

Wägeplattform:

Holzkonstruktion, Eisen verstärkte Ecken
4 Scher-Wägezellen
lineare Signalverstärkung
max. Belastung: 4000 kg
Genauigkeit: +-2kg

CODE-ENTWICKLUNG



```
public class MultiCastBehavior extends Behavior{

    private TransformGroup targetTG;

    // free parameter for fine tuning
    private float phiX      = 0.0f , phiY    = 0.0f;
    private float L         = 1.0f;
    private float offCenter = 0.85f;
    private float rotX      = 0.0f , rotY    = 0.0f;
    private float signX     = 1.0f , signY   = 1.0f;
    private float xOffset   = 0.0f , yOffset = 0.0f , zOffset = 0.0f;

    // data from multicast
    private float f1 = 0.0f,f2 = 0.0f,f1Prev,f2Prev;

    private WakeupCondition my_WakeupCondition = null;
    private MultiCastClient myClient;                  // Multicast Client
```

```
        translation.setTranslation(new Vector3f(0.0f,offCenter,0.0f));
        rotation.rotX(rotX+signX*phiX);
        rotation.mul(translation);
        rotation2.rotZ(rotY+signY*phiY);
        rotation2.mul(rotation);
        translation.setTranslation(new Vector3f(xOffset,yOffset-offCenter+phiX*L,zOffset));
        translation.mul(rotation2);
        targetTG.setTransform(translation);
        this.wakeupOn(my_WakeupCondition);
    }
}
public BranchGroup createSceneGraph() {
    // Create the root of the branch graph
```

Das Projekt „Loser Raum“ ist ein Beispiel für eine verteilte Computeranwendung im künstlerischen Kontext. Für jede Wandprojektion wird ein eigener Rechner und ein Beamer eingesetzt. Die vorproduzierte Wandfotografie ist als zweidimensionales Objekt in eine Java3D Umgebung eingebettet und kann über zwei Winkel gedreht und gekippt, bzw. dreidimensional transformiert (verschoben) werden (Java-Applikation „BalanceRoomDisplay“). An einen der Rechner ist die Wägeplattform angeschlossen. Auf diesem Gerät wird zusätzlich die physikalische Bewegung einer zweidimensionalen Wippe in einem viskosen Medium simuliert („BalanceRoomServer“). Über das angeschlossene Netzwerk (Ethernet) werden die Simulationsparameter (Neigungszustand der Wippe) an die Display-Applikationen gesendet, und es entsteht der Eindruck eines zusammenhängend animierten Raumes („low cost cave“).
Für die Java-Programmierung wurden CodeKit-Module aus den Bereichen „Netzwerk“ und „Java3D“ benutzt und erweitert. Weil das projizierte Bild mit der realen Wand – im Gleichgewichtszustand – exakt übereinstimmen muss, wurde ein „TuningServer“-Programm geschrieben. Die Java-Applikation ermöglicht die Feinjustierung der virtuellen Wände und kompensiert gleichsam Verzerrungsartefakte in der Wandfotografie, was den Aufbau der Installation überhaupt erst ermöglicht.
Wichtig ist uns, dass die in der aufwändigen Installation gesammelten Erfahrungen und Hilfsprogramme anderen Studierenden in ihren künstlerischen Projekten zur Verfügung stehen. Deshalb wurde die Software nachträglich ausführlich kommentiert, aufgearbeitet und steht im CodeKit-Archiv frei zur Verfügung. Fragemente dieser Programme wurden bereits für die Arbeit „Update (zo2)“ von Vera Doerk weiterverwendet.

Source-Code:

BalanceRoomDisplay.java
BalanceRoomServer.java
TuningServer.java

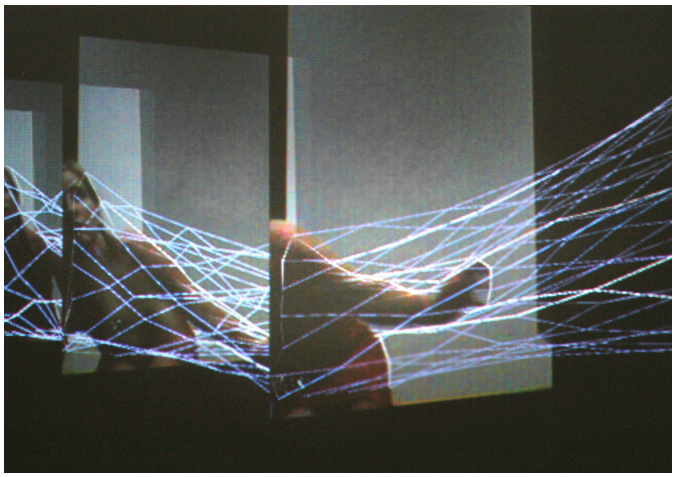
AUSSTELLUNGEN

Altitude 2002 [http://www.khm.de/altitude]
Ars Electronica 2002

LINKS

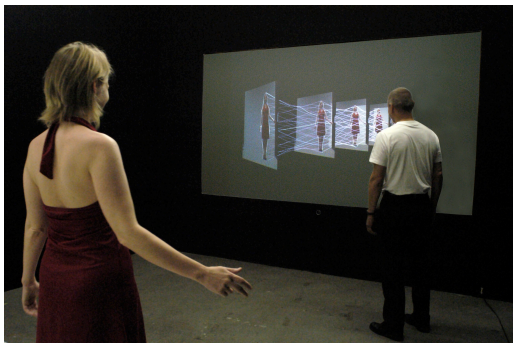
Digital Sparks Award [http://netzspannung.org/digital-sparks/]
Anja Kempe Homepage [http://www.khm.de/~akempe]

UPDATE (ZO2) – VERA DOERK



Update (zo2): Vernetzte Bildebenen

KURZBESCHREIBUNG



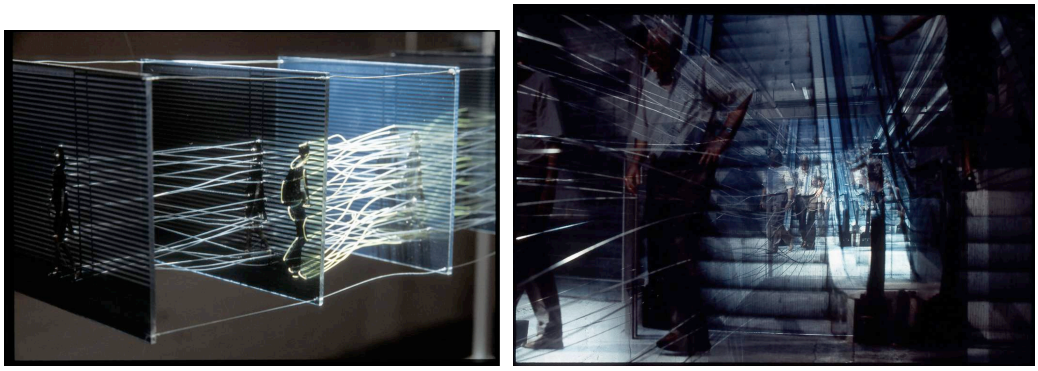
„Update (ZO2)“ ist konzipiert als weiterführende Arbeit der „Zeitobjekte“ und ist die Diplomarbeit von Vera Doerk, entstanden 2003 im Interface Labor der Kunsthochschule für Medien Köln. In einer interaktiven Installation werden die Konturen der Besucher erfasst, ihre Bilder auf zeitlich und räumlich getrennten Bildebenen wiedergegeben und Konturpunkte durch komplexe Netzstrukturen verwoben. Die Java-Programmierung wurde betreut von Jochen Viehoff.

„O-TON“ DER KÜNSTLERIN

„Bei der interaktiven computer-unterstützten Installation "Update (zo2)" steht der Betrachter seinem mit einer Kamera aufgenommenen Abbild in Echtzeit gegenüber.Die Bilder erscheinen auf der Projektionsleinwand als Bestandteil eines virtuellen, dreidimensionalen Computermodells. Dabei wird jeweils das erste Bild der fünf im Modell dargestellten Bilder viertelsekündlich aktualisiert. Bei jeder Aktualisierung werden die vorausgegangenen Bilder weiter nach hinten gereicht (update). Der Computer erkennt die Silhouetten der sich vor der Kamera befindlichen Personen und ermittelt daraus die Koordinaten für die Netzstrukturen

zwischen den Bildern. Die entstehenden Netzgebilde sind dabei einem ständigen Wandel unterworfen. Der Betrachter kann durch seine Position vor der Kamera die perspektivische Darstellung des Modells beeinflussen und es damit sowohl zur Seite als auch nach oben und unten drehen. Zusätzlich wird auch der Sound durch Bewegung gesteuert. Der Frequenzbereich und die Lautstärke kann verändert werden.“

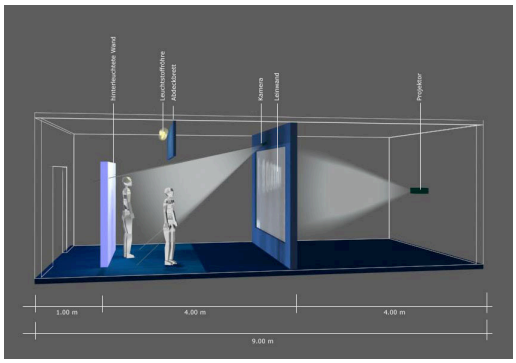
ENTSTEHUNGSGESCHICHTE



Zeitobjekte: „Hand verdrahtete“ Bildebenen

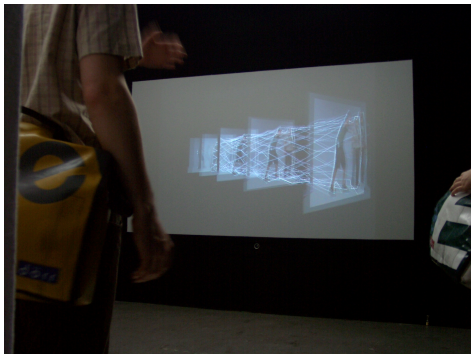
Die Arbeit „Zeitobjekte“ von Vera Doerk war anfänglich eine klassische Installation unter Verwendung „analoger“ Medien. Sie zeigte auf transparente Acrylplatten belichtete Fotografien unterschiedlicher, zeitlich separierter Personenkonstellationen. Die einzelnen Platten wurden hintereinander montiert und exponierte Konturpunkte der Personen durch Drähte verbunden. So entstanden handgefertigte Skulpturen: die „Zeitobjekte“.
Nachdem die Studentin im Rahmen des Java-Programmierkurses und weiterführender Fachseminare mit den Freiheiten und Möglichkeiten der Programmierung konfrontiert wurde und sich mit Java vertraut gemacht hatte, konkretisierte sie ihre Idee für einen „Update“ der „Zeitobjekte“ als interaktive (Computer) Installation. Wichtig für den Einstieg in die Programmierung war, dass für die Künstlerin CodeKit-Module bereitstanden, die einen Einstieg in die selbstständige Programmierung erleichterten, wenn nicht ermöglichten. Schrittweise kamen neue (Hilfs-)Programme hinzu auf dem Weg zu einer automatisierten Generierung von „Zeitobjekten“ in Echtzeit.
„Update (zo2)“ als studentische Arbeit demonstriert, wie sorgfältig betreutes Programmieren über einen längeren Zeitraum hinweg, gemeinsam mit großem Ehrgeiz der Studentin, zu eigener, groß angelegter künstlerischer Software führen kann.

HARDWARE, FUNKTION UND AUFBAU



Update: Installationsskizze

In der Installation „Update“ werden mit einer fest installierten Kamera die Personen in einem Raum aufgenommen. Zeitlich versetzt aufgenommene Einzelbilder werden in ein dreidimensionales Objekt als hintereinander angeordnete Bildebenen montiert. Ein Tracking-Algorithmus sucht gleichzeitig die Konturpunkte der Personen im Raum. Auf Grundlage dieser Eckdaten werden die Bildebenen in dem 3D-Objekt durch Linien verbunden. Abhängig vom Maß der Bewegung im Raum ist die Netzstruktur mehr oder weniger komplex ausgeprägt.



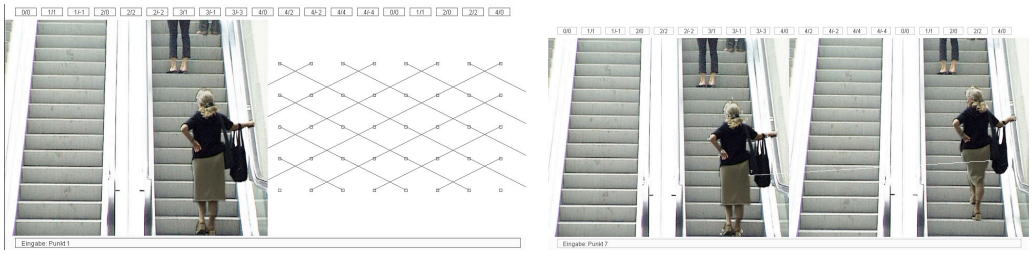
Die errechneten 3D Objekte werden über eine Rückprojektion auf die Leinwand projiziert und kontinuierlich erneuert. Ein leistungsfähiger PC analysiert das Live-Bild der Kamera, berechnet die Personenkonturen, generiert und animiert gleichzeitig die 3D Objekte und synthetisiert schließlich auch noch den Hintergrundklang der Installation.

CODE-ENTWICKLUNG

```
147    // ****************************************************
148    //    createSceneGraph
149    // ****************************************************
150
151    public BranchGroup createSceneGraph() {
152        // Create the root of the branch graph
153        BranchGroup objRoot = new BranchGroup();
154
155
156        // APPEARANCE Image
157
158        TextureLoader texqa = new TextureLoader(image01, TextureLoader.BY_REFERENCE,this);
159
160        texture = (Texture2D) texqa.getTexture();
161        texture.setCapability(Texture.ALLOW_IMAGE_WRITE);
162        texture.setCapability(Texture.ALLOW_IMAGE_READ);
163        Appearance apqa = new Appearance();
164        apqa.setCapability(Appearance.ALLOW_TEXTURE_WRITE);
165        apqa.setTexture(texture);
166
167        ic5  = (ImageComponent2D)texture.getImage(0);
168        ic5.setCapability(ImageComponent2D.ALLOW_IMAGE_READ);
169        ic5.setCapability(ImageComponent2D.ALLOW_FORMAT_READ);
170
171        bi = ic5.getImage();
```

Update: Auszug aus dem Java-Programm

Die Code-Entwicklung für die Arbeit lässt sich in drei Stufen unterteilen:
In der ersten Ausbaustufe des Projektes wurden fünf zweidimensionale Bilder zu einem dreidimensionalen Objekt (in Java3D) hintereinander angeordnet. Das entstehende Objekt ist transparent und kann (z.B. mit der Maus) animiert werden. Periodisch rotierend können die Bilder in dem Objekt ausgetauscht werden.



Update: Erzeugung der Netzstrukturen

Durch ein Java-Hilfsprogramm können – als zweite Ausbaustufe - die Koordinaten für die Kontur einer Person manuell erfasst und in die Java3D-Applikation integriert werden. Es entstehen Netzstrukturen zwischen den Ebenen, gemäß einer vorgegebenen Verknüpfungsordnung.

Schließlich werden die vorgefertigten Fotografien durch ein Live-Kamerabild ersetzt. Ein Tracking-System (Differenzverfahren) analysiert die Pixel-Informationen und berechnet in Echtzeit die Konturdaten der Personen („ImageGrabber“). Das Bild und die Konturdaten werden an die Java3D Applikation („ImageServer“ und „ImageClient“) gesendet und angezeigt („ImageViewer“) – es entstehen die Objekte mitsamt ihren Netzstrukturen von „Update“.
Abhängig von der Positionierung einzelner Personen im Raum wird das Zeitobjekt in der Projektion ausgerichtet und bietet unterschiedliche

Betrachtungswinkel. Ferner wird korrespondierend zu den Bewegungen ein Klang synthetisiert. Mehrere der in dieser Installation verwendeten Programme gehen auf CodeKit-Module zurück, wurden jedoch von der Künstlerin eigenständig so erweitert und umgeschrieben, dass ihre eigene interaktive Software daraus erwuchs.

Source Code:

ImageViewer.java
ImageServer.java
ImageClient.java
ImageGrabber.java

AUSSTELLUNGEN

Altitude 2003 [http://www.khm.de/altitude]
Pop Komm 2003 (Köln)
Tanzmedial 2003 (Köln) [http://www.sk-kultur.de/]

LINKS

Vera Doerk Homepage [http://www.khm.de/~veradoerk/]

BITMIRROR – TOBIAS GREWENIG

BitMirror: animierte ASCII Zeichen

KURZBESCHREIBUNG

Bitmirror: Vom Bild ... zum Zeichen

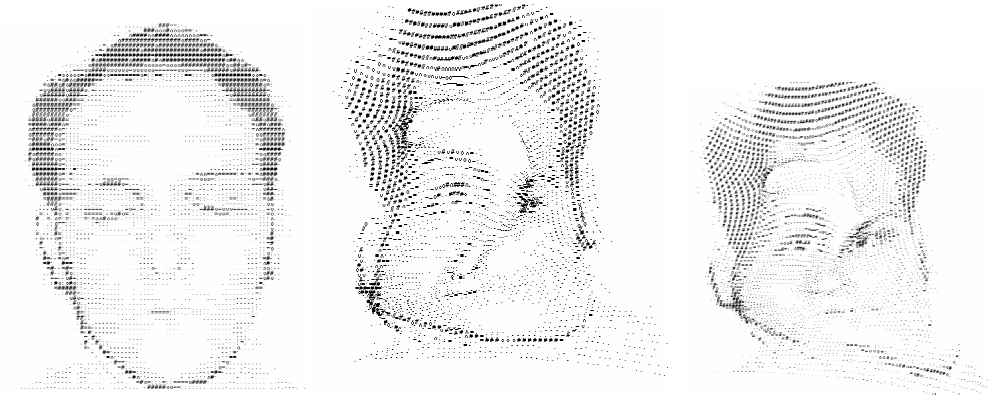
„Bitmirror“ ist eine aktuelle Projektarbeit (2003) von Tobias Grewenig die im Interface-Labor (lab3) als „work in progress“ weiterentwickelt wird. Die Pixel-Informationen aus Bilddateien oder von einer Web-Kamera werden in Ascii-Zeichen, sowie Töne übersetzt und gemäß den Regeln eines Vielteilchensystems durch physikalische Kräfte animiert, bzw. moduliert. Die Arbeit wurde vom Künstler selbst in Java und Pure Data programmiert.

„O-TON“ DER KÜNSTLERS

„Im Bitmirror sieht sich der Betrachter in seiner digitalisierten Form, aufgelöst in abstrakten Partikeln und akustischen Fragmenten.
Die Darstellung ist Ascii-Code, der zu den puristischen und ältesten Stilmitteln der Komputergrafik zählt.

Analog zur Grafik wird der akustische Input durch Granularsynthese in Partikel zerlegt.
Bewegung, Geräusche und Stimme lösen in diesem grafischen und akustischen Partikelsystem ein simuliertes Schwarmverhalten aus.
In dieser Arbeit habe ich versucht, ein Echtzeitsystem zu entwickeln, das mit Hilfe dieser Darstellungsformen und Algorithmen den Kontrast herauskristallisiert zwischen dem reellen, analogen Abbild eines Menschen und der sterilen, abstakten, digitalen Form des Komputers.“

ENTSTEHUNGSGESCHICHTE



Bitmirror: Die Zeichen kommen in Bewegung

Tobias Grewenig ist postgraduierter Student an der Kunsthochschule für Medien und hat aus seiner vorhergehenden Ausbildung fundierte Kenntnisse in der Programmierung multimedialer Inhalte für seine Arbeit an der KHM mitgebracht. Während des Java-Programmierkurses – ein zweiwöchiger Kompaktkurs zu Beginn des Sommersemesters 2003 – konnte er sein Vorwissen nutzen, um rasch in die objektorientierte Programmiersprache Java einzusteigen. Die Inhalte des Kurses – insbesondere Grundlagen der Partikelanimation - deckten sich mit den späteren künstlerischen Experimenten des Künstlers, so dass er die grundlegenden Komponenten des „Bitmirrors“ aus den CodeKit-Bausteinen übernehmen und ausbauen konnte. Wichtig war in diesem Zusammenhang, dass der Student früh die theoretischen Grundlagen verstehen konnte, etwa die Berechnung von Kräften in einem Vielteilchensystem, oder die Integration der Newton-Bewegungsgleichungen.
Eine zweite Entwicklungsphase des Projektes konzentrierte sich auf elementare Tracking-Algorithmen und die Auswertung eines Webcam-Bildes. Die Bewegungen eines Betrachters werden interpretiert und steuern die Animation, lenken die Zeichenströme und zeichnen sich verantwortlich für die Dynamik des Prozesses.
In einer dritten Entwicklungsstufe wurde die Dynamik des Vielteilchensystems in eine Audioapplikation integriert. Tobias Grewenig nutzt die freie Programmierumgebung „Pure Data“ für die Umsetzung von Bewegung in Klang.
Zusammenfassend kann die Projektarbeit „Bitmirror“ als eigenständige Programmierarbeit angesehen werden, die eine Betreuung zwar nicht überflüssig gemacht hat, sich aber auf ein minimales Maß beschränkte, nämlich auf Anregungen zur Code-Gestaltung, Aufzeigung von Möglichkeiten und Fehlersuche. Die Partikelanimation des „Bitmirrors“ basiert auf dem CodeKit-

Modul „ParticleAnimation“. Der „Bitmirror“ ist ein Beispiel für die selbstständige experimentelle Arbeit im Rahmen fortschrittlicher Programmierung und das Projekt zeigt, wie Ausbildung, Betreuung und künstlerische Arbeit miteinander verflochten sein können.

HARDWARE, FUNKTION UND AUFBAU

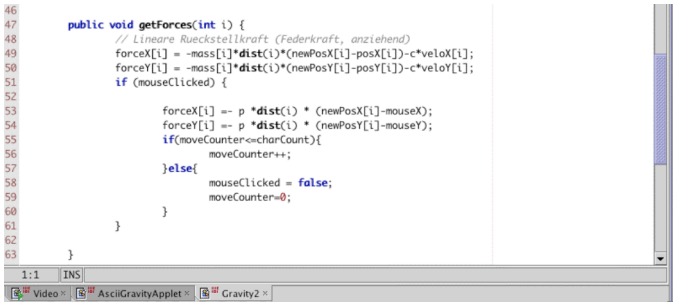
Bitmirror: Schwarmverhalten der animierten Ascii-Zeichen

Das „Bitmirror“ Programm läuft auf jedem Betriebssystem, auf dem eine Java 2 Virtuelle Maschine (JVM) installiert ist. Aufgrund der hohen erforderlichen Rechenleistung, sind schnelle Computer erforderlich (ab 2.0GHz).
Der Aufbau besteht aus einem Rechner und einer Webcam, die einen Betrachter des „Bitmirror“ aufzeichnet. Die Bewegung des Betrachters steuert die Animation der Ascii-Zeichen sowie die parallel ablaufende Klangsynthese. Über Tastenkombinationen kann das Bild invertiert, vergrößert oder verkleinert, oder in den Ursprungszustand (ohne Gravitationskräfte) zurückgesetzt werden. Weitere Hard- oder Software ist nicht erforderlich.

CODE-ENTWICKLUNG

```
64
65      public void iterateGauss(int i) {
66              getForces(i);                           // Berechnung der neuen Kraefte
67
68              veloX[i] += dt*forceX[i];              // neue Geschwindigkeiten
69              veloY[i] += dt*forceY[i];
70              veloX[i]*=dec[i];
71              veloY[i]*=dec[i];
72              newPosX[i]  += dt*veloX[i];             // neue Positionen
73              newPosY[i]  += dt*veloY[i];
74              //        dec[i]*=0.999;
75              //        dec[i]*=1.001;
76      }
77
78      public double dist(int i) {
79              return Math.sqrt((newPosX[i]-posX[i])*(newPosX[i]-posX[i])+
80                      (newPosY[i]-posY[i])*(newPosY[i]-posY[i]));
81      }
82 }
83
84
   1:1    INS
 Video ×  AsciiGravityApplet ×  Gravity2 ×
```

Bitmirror: Auszug aus dem Java-Programm



```
46
47      public void getForces(int i) {
48              // Lineare Rueckstellkraft (Federkraft, anziehend)
49              forceX[i] = -mass[i]*dist(i)*(newPosX[i]-posX[i])-c*veloX[i];
50              forceY[i] = -mass[i]*dist(i)*(newPosY[i]-posY[i])-c*veloY[i];
51              if (mouseClicked) {
52
53                      forceX[i] =- p *dist(i) * (newPosX[i]-mouseX);
54                      forceY[i] =- p *dist(i) * (newPosY[i]-mouseY);
55                      if(moveCounter<=charCount){
56                              moveCounter++;
57                      }else{
58                              mouseClicked = false;
59                              moveCounter=0;
60                      }
61              }
62
63      }
```

1:1 INS

Video × AsciiGravityApplet × Gravity2 ×

Bitmirror: Auszug aus dem Java-Programm

Die Code-Entwicklung des „Bitmirror“ basiert – bezüglich der Partikelanimation – auf dem CodeKit-Modul „ParticleAnimation“. Als Auszug dargestellt sind die zwei wesentlichen Routinen aus dem Modul, erstens die Berechnung der Kräfte (getForces) zwischen den einzelnen Buchstaben, und zweitens die Integration der Bewegungsgleichungen (iterateGauss), die Grundlage für eine realistisch anmutende Dynamik der Bewegung (z.B. Wellenbewegungen, Schwarmverhalten usw.).
Als Zeichensätze wurden die Java 2 Fonts und ein Standard Graphics2D als Zeichenfläche verwendet, jeder Partikel in dem System wird durch ein Zeichen repräsentiert.
Ein einfacher Tracking-Algorithmus wertet die Pixel-Informationen der Kamera in Echtzeit aus (Differenzbild-Verfahren, z.B. CodeKit-Modul SimpleTracking [SimpleTracking.java]). Grundlegend dafür ist die Quicktime-Erweiterung „VideoByPixel“. Das verwendete Packet „vbp.jar“ und einige Tracking-Demoprogramme finden sich in der CodeKit-Rubrik „KameraTracking“ und „VideoGrabbing“. Sie ermöglichen schnelles „offscreen grabbing“ in Echtzeit.

Source-Code (AsciiGravityApplet):

AsciiGravityAppet.java
Gravity2.java

AUSSTELLUNGEN

Altitude 2003 [http://www.khm.de/altitude/]

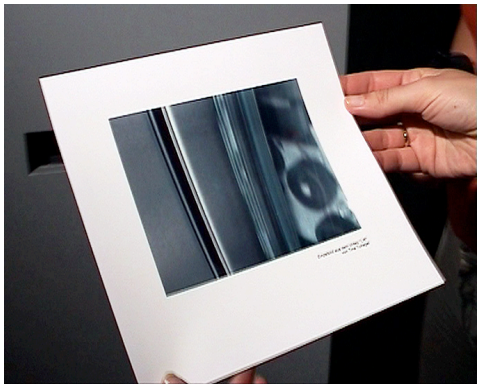
LINKS

Tobias Grewenig Homepage [http://www.khm.de/~xi-bot/]

FRAMEDRUCKER – TINA TONAGEL

FrameDrucker: Verkauf eines Videobildes

KURZBESCHREIBUNG



„Framedrucker“ ist ein studentisches Projekt von Tina Tonagel und entstand 2003 im Interface-Labor (lab3) der Kunsthochschule für Medien Köln. Der Münzautomat ermöglicht Auswahl und Kauf von Einzelbildern aus einem projizierten Video. Die „gekauften“ Frames werden auf Fotopapier gedruckt und aus dem Film gelöscht – die verbleibende Videosequenz ist um genau ein Frame kürzer. Die Programmierung basiert auf modifizierten Komponenten des CodeKit-Archivs und wurde von Jochen Viehoff betreut. Martin Nawrath entwickelte die erforderliche Interface-Hardware.

„O-Ton“ der Künstlerin

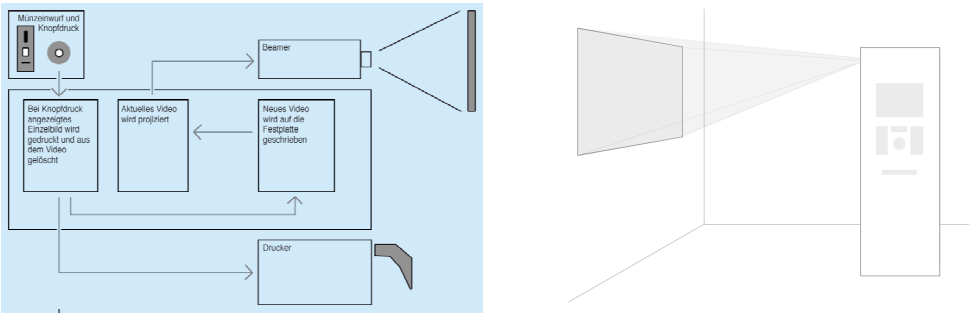
„Der Framedrucker ist ein Automat, der die Dekonstruktion eines Videos durch den Kauf einzelner Frames ermöglicht:
Während ein (speziell für diesen Zweck produziertes) Video als Loop an eine Wand projiziert wird, hat der Betrachter die Gelegenheit, Einzelbilder dieses Videos käuflich zu erwerben.
Nach Münzeinwurf kann das Video durch Knopfdruck angehalten werden.
Das in diesem Moment angezeigte Bild wird ausgedruckt und daraufhin aus dem Video gelöscht.
Das Video löst sich auf; es wird immer kürzer (die verbliebenen Frames rücken zusammen; die Lücke wird geschlossen), bis es - komplett in seine Bestandteile zerlegt - im ursprünglichen Medium nicht mehr existiert.“

Entstehungsgeschichte

Dass Videos interaktiv gesteuert und verändert werden können, wird als Lehrinhalt im Java-Programmierkurs und in Fachseminaren thematisiert. Unterschiedliche Beispiele dafür werden im CodeKit-Programmarchiv unter der Rubrik „VideoControl“ permanent weiter entwickelt und ergänzt (s. auch Code & Interaktion 2): „ThermoVideo“, eine Temperatursteuerung für Videofilme). In diesem Umfeld entstehen immer wieder studentische Arbeiten im Interface-Labor, die sich mit dem Medium „interaktiver Film“ auseinander setzen. Der „FrameDrucker“ von Tina Tonagel ist ein Beispiel dafür. Die Idee der Künstlerin, einen Automaten für die „Dekonstruktion“ des Videofilms zu entwerfen, wurde durch Betreuung der (Java-)Programmierung und der (Interface) Elektronik-Komponenten ermöglicht.
Wichtig war es der Künstlerin, dass die verkauften Frames auch tatsächlich aus dem Original unwiderbringlich gelöscht werden mussten. Der „FrameDrucker“ ist kein „Fake“-Objekt, jede einzelne Komponente, vom Programm bis zur Münzprüferelektronik sollte voll funktionsfähig und für den Käufer nachvollziehbar sein.
Der Programmieraufwand selbst war überschaubar, weil im Wesentlichen Module aus dem CodeKit-Archiv benutzt und erweitert wurden. Viel Mühe und Liebe

zum Detail investierte die Künstlerin in die Gestaltung und den Holzbau des Automaten.

HARDWARE, FUNKTION UND AUFBAU



FrameDrucker: Schematischer Aufbau

Im Inneren des Automaten befindet sich ein PC, auf dessen Festplatte das Video gespeichert ist. Eine auf „Quicktime for Java“ basierende Java-Applikation (s.u.) spielt das Video in einer Endlosschleife ab, die Wandprojektion erfolgt durch einen ebenfalls in dem Automaten installierten Daten-Beamer.
Ein elektronischer Euro-Münzprüfer nimmt das Geld entgegen – derzeit 4 Euro pro Ausdruck. Das Gerät erkennt alle Euro-Münzen und ist so programmiert, dass beliebige Kombinationen bis zum angegebenen Betrag addiert werden können. An den Münzprüfer angeschlossen ist eine LCD-Anzeige, die den Käufer schrittweise anleitet und den verbleibenden Restbetrag anzeigt.
Nach Einwurf von 4 Euro hat der Betrachter beliebig lange Zeit, um sich das Video anzuschauen und schließlich den „Auslöseknopf“ zu drücken, um den Druckvorgang zu starten.
Die Java-Applikation hält das Video an, so dass das ausgewählte Bild für etwa 12 Sekunden als Standbild zu sehen ist. Anschließend läuft das Video normal weiter, allerdings ohne das verkaufte Einzelbild.
Nach 90 Sekunden ist das Bild gedruckt und kann dem Auswurfschlitz entnommen werden.
Das verwendete Papier hat ein Format von 21 x 20 cm, das Bild ist mittig in der Größe 14 x 11 cm aufgedruckt, darunter steht als Signatur: Einzelbild aus dem Video „Lift“ von Tina Tonagel.
Der Münzprüfer zählt außerdem unabhängig vom Geldeinwurf die Anzahl der Ausdrucke und zeigt im Display an, wenn das Druckerpapier verbraucht ist.

CODE-ENTWICKLUNG

```
139    // ************************************************
140    //  Print one Frame
141    // ************************************************
142    public void printFrame()  {
143
144        try {
145            mv.setRate(0);                                          // stop movie
146
147            bi   = pixelRobot.createScreenCapture(screenCapture);   // grab pixels from screen & cre
148            pixel = bi.getRGB(0,0,width,height,pixel,0,width);
149            img = createImage(new MemoryImageSource(width,height,pixel,0,width));
150
151            PrintResolution pr = new PrintResolution(img,dpi,false);   // print image via PrintResoluti
152            pr.print();
153            pr = null;
154
155            mv.deleteSegment(mv.getTime(),tFrame);     // cut frame
156            saveMovie();                               // save movie
157            mv.setRate(1);                             // continue movie
158        }
159        catch (Exception qte){System.out.println(qte);}
160
```

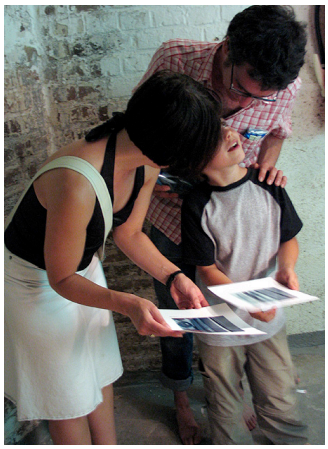
FrameDrucker: Auszug aus dem Java-Programm

Die Software für den „FrameDrucker“ verwendet Standardmodule des CodeKit-Archivs zum Abspielen von Quicktime-Videos und der Ansteuerung des seriellen Ports (com1 oder com2). Das Hauptprogramm ist die ausführbare Java-Applikation „Video“, die das Video in einer Endlosschleife im „FullScreen-Modus“ abspielt. Die Kommunikation mit dem Euro-Münzprüfer wird durch die Klasse „Muenzpruefer“ realisiert. Für den Ausdruck eines Frames auf den (USB) Thermosublimationsdrucker wird die Klasse „PrintResolution“ benutzt. In dem gezeigten Auszug des originalen Java-Codes wird ersichtlich, dass die verbleibende Videosequenz – nach dem Löschen eines Frames – neu abgespeichert wird. Dieser Prozess dauert einige Sekunden und überschreibt die alte Version des Videos. Die Originalversion kann nicht wiederhergestellt werden.

Source Code:

Muenzpruefer.java
Video.java
PrintResolution.java

AUSSTELLUNGEN & PREISE



Altitude 2003 [http://www.khm.de/altitude/]
Medienwinter 2004 (Stuttgart) [http://www.filmwinter.de/]
Milla und Partner Preis „Medien im Raum" [http://www.filmwinter.de/]

LINKS

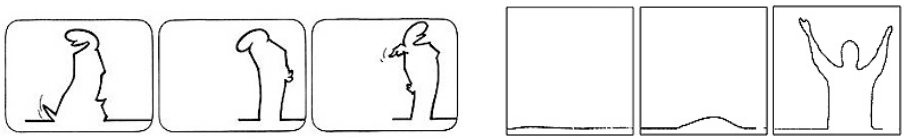
Tina Tonagel Homepage [http://www.khm.de/~ttonagel]

STECKBRIEFE

Projektpräsentation im Interface-Labor (lab3)

Alle im „Steckbrief" zusätzlich kurz vorgestellten Projekte sind im lab3 (Interface-Labor) an der Kunsthochschule für Medien Köln entwickelt worden und benutzen und basieren auf den frei zugänglichen Java-Programmen des CodeKit-Archivs [http://java.khm.de].

THELINE

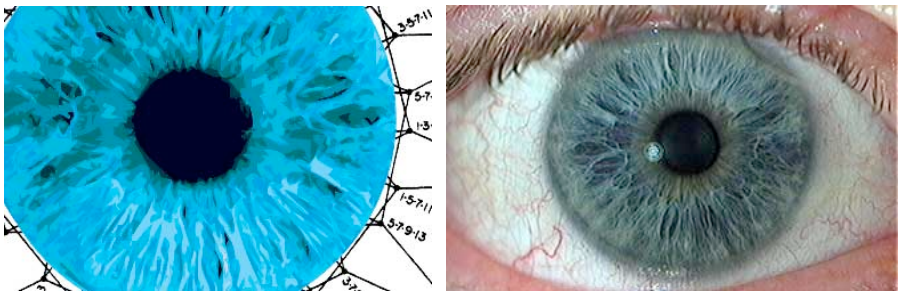


Oswaldo Cavandoli: La Linea ... Interaktive Installation: TheLine

Interaktive Installation von Mone Kante und Jochen Viehoff (2001)
Die Kontur einer Person steuert - in Anlehnung an Oswaldo Cavandolis "La Linea" - durch Gravitationskräfte eine physikalisch animierte, zusammenhängende Linie.

Partikelanimation in Echtzeit ("Perlenkette")
Kamera-Tracking (Schwellwertanalyse)
Software: TheLine.java

BIOMETRIK SOUND ENGINE

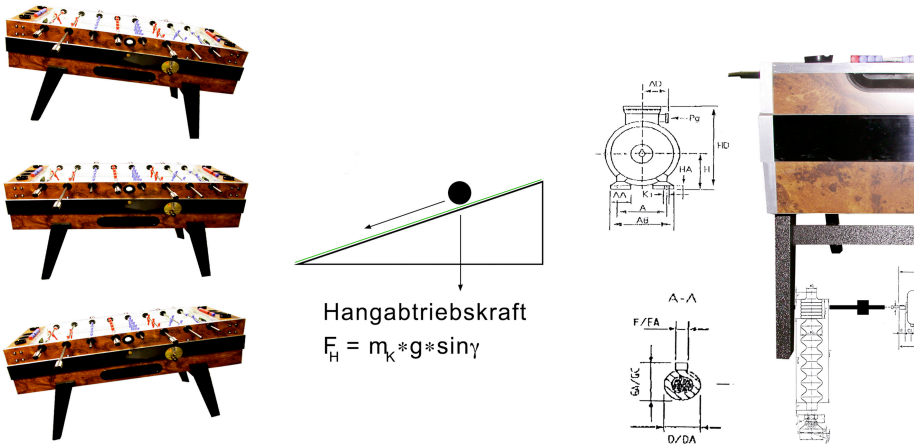


Biometrik Sound Engine: Farbwerte für die Klangerzeugung vom menschlichen Auge

Installation von Volker Morawe (2002)
Die Iris – der farbige Teil des menschlichen Auges - wird in Rotation versetzt, während ein „Tonabnehmer" die Farbinformationen in Klänge umwandelt.

Biometrik Sound Engine Homepage [http://www.datenpunpe.de]
Software: RotateImage.java, ImagePlayer.java

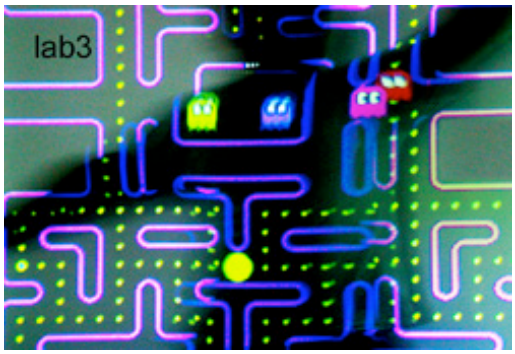
KIPPKICK

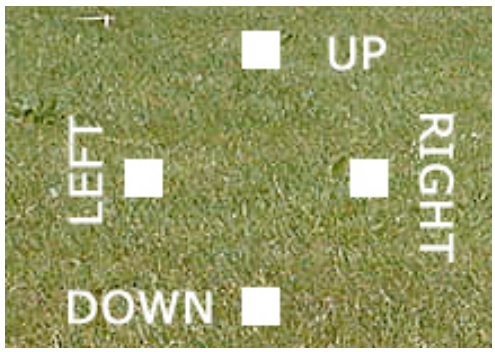


Interaktive Installation, Seminararbeit (2002)
Die Zuschauer steuern durch ihre Positionierung im Umfeld eines Tischfußballspiels (Kicker) dessen physikalische Neigung. Gegen die Gunst des Publikums spielt die unbeliebte Mannschaft bergauf.

Tischfußball, elektrischer Antrieb für variable Neigung (+-10 Grad)
Kamera-Tracking (Differenzbildanalyse)
Software: KippKick.java, KippKickControl.java

PLAYCOURT





Interaktive Installation, Seminararbeit (2002)
Der ganze Körper wird auf einem Spielfeld als Fernsteuerung für Computerspielklassiker (z.B. Pacman, Tetris usw.) genutzt. Gruppen können sich unterschiedliche Steuerungsaufgaben teilen.

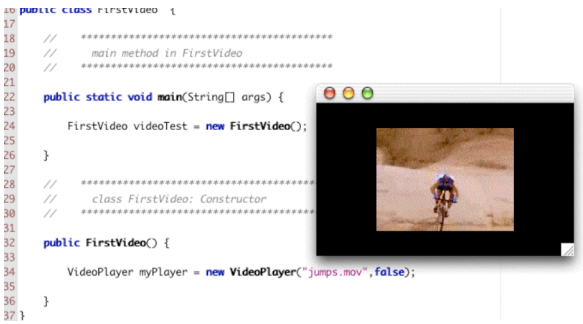
Spielfeld mit vier „sensiblen" Tasten
Kamera-Tracking (Schwellwertanalyse)
Software: PlayCourt.java, RemotePacman.java

RUNDE 10 UND BOXINGBATTERIE

Interaktive Installation, Jochen Viehoff (2002)
Der interaktive KHM Boxsack wurde für die Entwicklung unterschiedlicher Spiele und Anwendungen genutzt.

Boxsack: Druck- und Beschleunigungssensoren
Runde 10 – ein Boxwettkampf
BoxingBatterie – neue Ausdauer am Schreibtisch
Software: BoxingBatterie.java, Runde10.java
Sound Slam Homepage [http://www.fursr.com/]

CODEKIT – TUTORIAL



CodeKit: Multimediale Programmmodule für die künstlerische Projektarbeit

INTENTION

Das CodeKit-Tutorial ist eine kurze, schrittweise Einführung in das selbstständige Arbeiten mit Programmmodulen aus dem CodeKit-Archiv. Das Tutorial soll als Grundlage für eigene, weiterführende Experimente dienen, kann jedoch einen Java-Programmierkurs nicht ersetzen.

INSTALLATION

Im ersten Abschnitt wird die Installation von Java 2 und der integrierten Entwicklungsplattform NetBeans (Sun) für Windows (PC) und OS X (Macintosh) beschrieben. Ferner werden die Erweiterungen „Quicktime for Java“ und – nur für den PC – die „communication API“ zur Ansteuerung des seriellen Ports installiert, um den vollen Umfang der CodeKit-Programme nutzen zu können.

GETTING STARTED

Ausgehend von einer funktionierenden Java 2 Entwicklungsumgebung wird ein erstes einfaches CodeKit-Programm als Java-Applet geladen und gestartet, typische (Anfänger)-Fehlermeldungen werden aufgelistet.

CODE & INTERAKTION 1

Das CodeKit-Modul „SoundMixer“ wird vorgestellt und die wichtigsten Programmzeilen erklärt. Die Java-Applikation ermöglicht das Mischen von zwei Musikstücken durch einfache Maus-Interaktion. In Übungen wird, ausgehend vom ursprünglichen Code, das Programm modifiziert und erweitert.

CODE & INTERAKTION 2

Die Java-Applikation „ThermoVideo“ ist ebenfalls Bestandteil des CodeKit-Archivs und steuert die Geschwindigkeit eines Videos über die gemessene Raumtemperatur. Für die Messung werden ein LabPro-Messcomputer (PC) und ein Vernier-Temperatursensor benötigt. Als Übungsaufgabe für Experten kann der Anwender oder die Anwenderin in dem Programm „ThermoVideo“ die Temperaturmesssung durch die Maus-Steuerung aus „SoundMixer“ ersetzen.

INSTALLATION

Freie Software für das CodeKit: Java und die integrierte Entwicklungsumgebung NetBeans

EINLEITUNG

Um das CodeKit-Archiv nutzen zu können, müssen unterschiedliche zusätzliche Software-Komponenten auf dem Rechner installiert sein. Grundlegende Kenntnisse im Umgang mit dem Computer werden hier vorausgesetzt (z.B. Anlegen neuer Ordner, Download von Dateien aus dem Internet etc.). Im Folgenden wird schrittweise die Einrichtung einer Java 2 Arbeitsumgebung, basierend auf der integrierten Entwicklungsplattform (IDE) NetBeans, skizziert. Als Erweiterungen sind zusätzlich „Quicktime for Java“ und die „communications API“ von Sun notwendig.

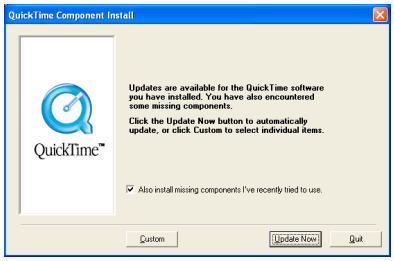
WINDOWS

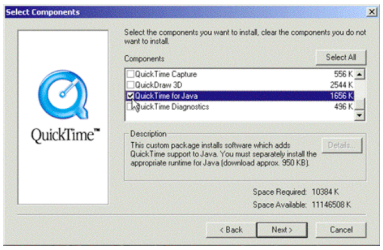
Die Java 2 Standard Edition (J2SE) in der aktuellen Version kann gemeinsam mit der integrierten Entwicklungsumgebung „Studio One“ (ehemals „Forte for Java“) kostenlos von der Sun-Homepage heruntergeladen werden:

Java2 & NetBeans IDE co-bundle (97MB) „Windows.exe“
[http://java.sun.com/j2se/1.4.2/download.html]
Installationsanleitung [http://java.sun.com/j2se/1.4.2/install_j2se142-nb351_windows.html]

Quicktime for Java

Die Installation des kostenlosen Quicktime Players **mit** den Komponenten für „Quicktime for Java“ kann unter „Custom Installation“ ausgewählt werden:





Installer Quicktime Player
[http://www.apple.com/quicktime/download/]
Custom Installation
Quicktime for Java Komponenten auswählen
Quicktime Installieren (oder Update)

Communications API

Für den Anschluss von LabPro-Sensoren an den seriellen Port ihres PCs wird die Java-Erweiterung „communications API“ benötigt:

Download „communications API“
[http://java.sun.com/products/javacomm/downloads/index.html]
Installationsanleitung: Readme.html

Die seriellen Anschlüsse „com1“ und „com2“ werden in Java-Applikationen erkannt, wenn die Datei „javax.comm.properties“ in das richtige Verzeichnis kopiert wurde.

MACINTOSH OS X

Mac OS X

Anmerkung: Java 2 Standard Edition (J2SE) ist bei dem Betriebssystem OS X vorinstalliert und steht für ältere Macintosh Betriebssysteme **nicht** zur Verfügung.

NetBeans IDE

Download Englisch Installer Mac OS X (28,7MB)
[http://www.netbeans.org/downloads/ide/index.html]
Das digital image (.dmg) enthält den NetBeans Launcher
Starten von Netbeans: File/Launch Netbeans

Quicktime for Java

Das Erweiterungspaket Quicktime for Java ist bei OS X vorinstalliert.

Die comminactions API ist **nicht** auf Macintosh OS X implementiert. Das CodeKit-Archiv wird in Zukunft den preiswerte USB-Sensoren unterstützen, die kompatibel sowohl für Windows als auch für Macintosh Rechner sind.

Im nächsten Abschnitt „Getting Started“ wird die NetBeans-Entwicklungsumgebung mit einem ersten Beispielprogramm aus dem CodeKit-Archiv getestet.

LINKS

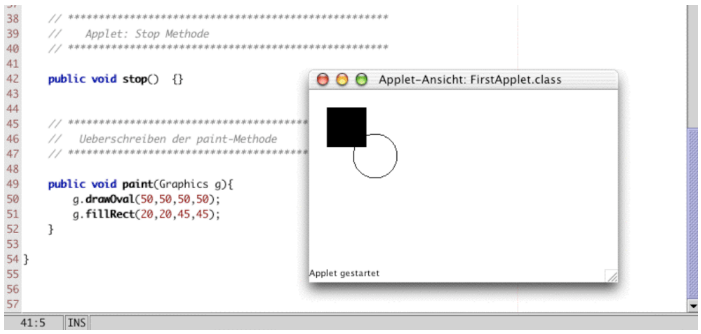
Java homepage [http://java.sun.com]

Apple Quicktime Homepage [http://www.apple.com/quicktime/products/]

communications API Homepage [http://java.sun.com/products/javacomm/]

NetBeans homepage [http://www.netbeans.org]

GETTING STARTED



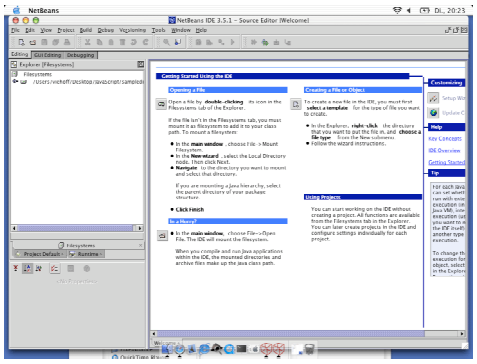
CodeKit: Das erste Java-Applet zeichnet zwei Objekte

EINLEITUNG

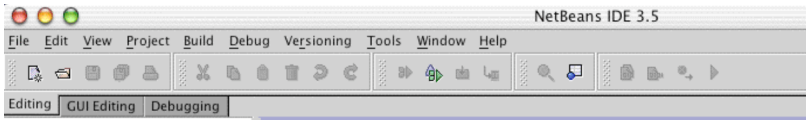
In dem Abschnitt „Getting Started“ wird schrittweise erklärt, wie die Entwicklungsumgebung NetBeans einzurichten ist, damit Java-Applets und Applikationen aus dem CodeKit-Archiv geladen und „compiliert“ werden können. Als „Compilation“ bezeichnet man die Übersetzung des Source-Codes in den sogenannten „Byte-Code“, der von der Java Virtuellen Maschine (JVM) auf dem jeweiligen Computer ausgeführt werden kann. Eine Auflistung der typischen Fehlermeldungen und weiterführende Links sollen über die üblichen Startschwierigkeiten hinweghelfen.

NETBEANS: EINRICHTEN DER ARBEITSUMGEBUNG

Startkonfiguration von NetBeans nach der Installation und dem Starten des Programms (zur Erinnerung: unter Windows heißt das Programm „Studio One“):

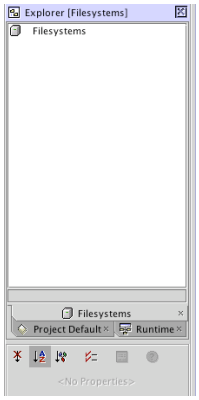


Die wichtigsten Fenster sind:



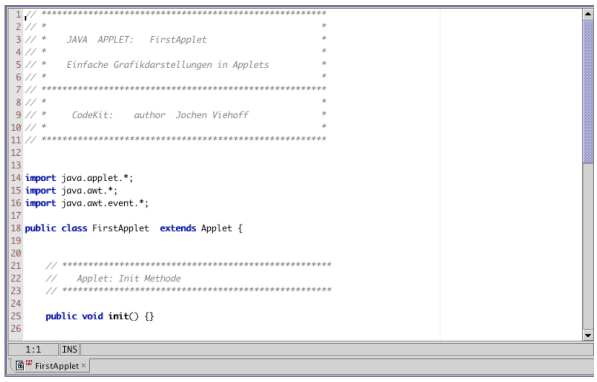
NetBeans: Befehlszeile

Befehlszeile von NetBeans: Compilieren, Starten von Programmen, Wechsel zwischen dem Modus „Editing“, „GUI Editing“, „Debugging“ und (später) „Running“.



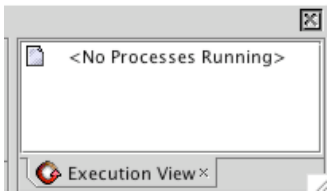
NetBeans: Explorer

Explorer: Verwaltet das File-System von NetBeans. Hier werden alle Erweiterungen (API) und die Verzeichnisse mit den Source-Codes eingetragen. Die Eigenschaften der angewählten Dateien werden im unteren Teil angezeigt und bei Bedarf verändert.



NetBeans: Source Editor

Source Editor: Hier werden die Codes editiert. Man kann zwischen den aktiven Dateien hin- und herspringen.



NetBeans: Execution Window

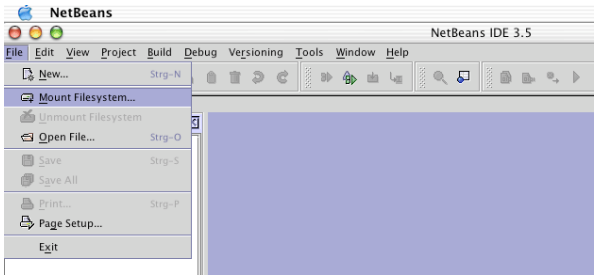
Execution Window: hier werden die laufenden Programme angezeigt und können manuell beendet werden.

NetBeans: Output Window (Konsole)

Output Window: Ausgabefenster für den Compiler (Editing-Mode) bzw. Ausgabekonsole im Running-Mode.

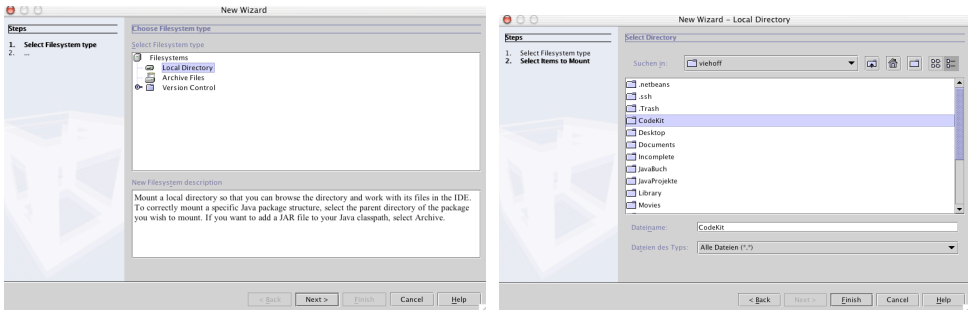
Mount Files System

Legen Sie ein neues, leeres Verzeichnis an, z.B. „CodeKit“.
Tragen sie das Verzeichnis „CodeKit“ in das File-System von NetBeans mit dem Befehl „File/Mount Filesystem“ ein.



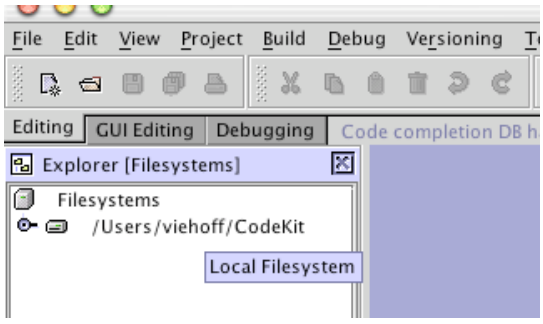
Mount Filesystem: Arbeitsverzeichnis anmelden

Wählen Sie „local directory“ und „next“. Wählen Sie das „CodeKit“ Verzeichnis aus (einmal anklicken) und drücken Sie „finish“.



Mount File System: Lokales Verzeichnis laden

Im NetBeans-Explorer erscheint das neue Verzeichnis:

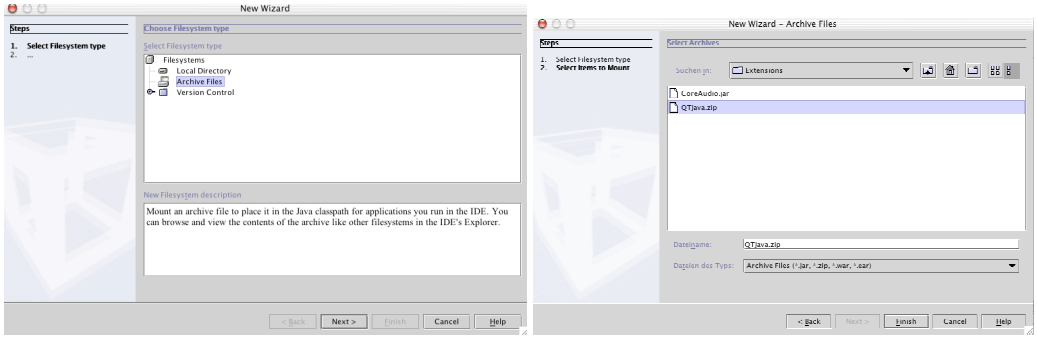


NetBeans: Arbeitsverzeichnis im Explorer

Mount Archive Files

Die Erweiterungen „Quicktime for Java“ bzw. „communication API“ müssen ebenfalls im File-System als Archive aufgelistet sein.

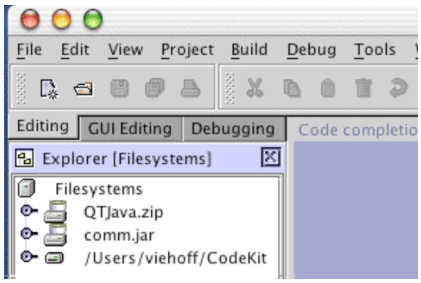
Führen Sie den Befehl „File/Mount Filesystem“ aus.
Wählen Sie „Archive Files“:



Mount File System: Archiv einbinden

Laden Sie folgende Archive (Suchen Sie die Position der Dateien mit der Such-Option auf Ihrer Festplatte):

Quicktime for Java: „QTJava.zip“
communication API (nur PC): „comm.jar“

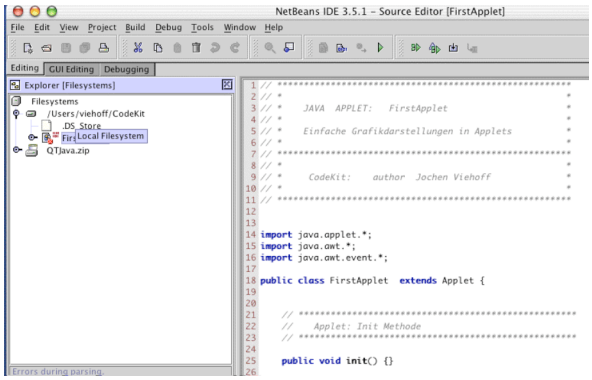


NetBeans: Archiv-Dateien und Arbeitsverzeichnis im Explorer

Werden die Archive-Files und das Arbeitsverzeichnis im NetBeans-Explorer angezeigt, ist die Einrichtung der Arbeitsumgebung abgeschlossen.

CodeKit: Herunterladen von FirstApplet

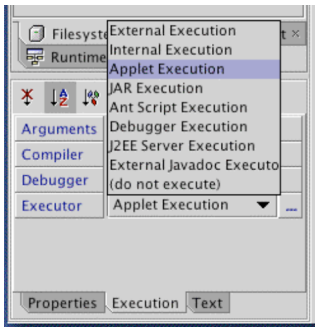
Laden Sie die Datei FirstApplet.java und speichern sie das Programm in ihrem Verzeichnis „CodeKit“. Im NetBeans-Explorer wird die Datei angezeigt und kann durch Doppelklick im Source-Editor geöffnet werden:



NetBeans: Öffnen des Source Codes im Editor

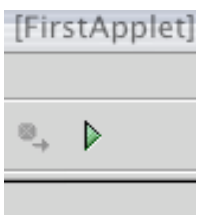
Compilieren und Starten

„FistApplet“ ist, wie der Name schon sagt, ein Java-Applet und kann als solches nicht als Programm ausgeführt werden, sondern muss auf einer html-Seite eingebunden werden. NetBeans erzeugt selbstständig eine passende html-Seite, wenn die Eigenschaften der Datei richtig eingestellt sind. Wählen Sie im NetBeans-Explorer „FirstApplet“ an (Einfach-Klick), die Eigenschaften werden angezeigt. Wählen Sie unter „Properties/Execution“ nicht „external Execution“ (Java-Applikation), sondern „Applet Execution“ aus:



NetBeans: Applet Execution auswählen

Compilieren und starten Sie das Java-Applet mit der Taste:



NetBeans: Java Programm compilieren und starten

Übung 1:

Zeichnen Sie zwei oder mehr Kreise, Boxen, indem Sie weitere Befehle mit unterschiedlichen Parametern hinzufügen, z.B.:

```
45    // ***************************************************
46    //   Ueberschreiben der paint-Methode
47    // ***************************************************
48
49    public void paint(Graphics g){
50        g.drawOval(50,50,50,50);
51        g.fillRect(20,20,45,45);
52        g.fillRect(100,100,30,30);
53        g.fillRect(100,150,45,45);
54    }
```

FirstApplet: Weitere Objekte zeichnen

Verstehen Sie die Funktionen der Parameter in den Befehlen? Versuchen Sie mit den schwarzen Boxen ein Schachmuster zu zeichnen.

CodeKit: Java-Applet FirstMelody

Speichern Sie folgende Dateien in ihr „CodeKit“ Verzeichnis (die Dateien sind auch im CodeKit-Archiv verfügbar)

FirstMelody.java
MidiInstrument.java

Öffnen Sie das Programm (FirstMelody.java) im Source-Editor. Starten Sie das Programm. Da es ein Java-Applet ist, müssen Sie die voreingestellten Eigenschaften ändern: Properties „Execution/AppletExecution“. Das Abspielen von (Midi)-Noten wird durch ein Objekt der Klasse „MidiInstrument“ realisiert. Die Klasse muss sich im File-System befinden und wird – bei Bedarf – von NetBeans selbstständig compiliert.
Sie hören einige Töne? Herzlichen Glückwunsch! Fangen Sie an, ein wenig zu komponieren...

Übung 2:

Untersuchen Sie den Source Code von FirstMelody.java und spielen Sie weitere Noten, indem Sie mehrere playNotes Befehle hinzufügen (cut&paste), z.B.:

```
32    // ***************************************************
33    //    Applet: Start Methode
34    // ***************************************************
35
36    public void start() {
37
38        //    play(note,volume,tonlaenge in0 msec)
39        piano.play(68,127,1000);                    // play note on piano (68: C')
40        piano.play(72,127,500);
41        piano.play(70,127,1000);
42        piano.play(58,127,1000);
43        piano.play(62,127,500);
44        piano.play(68,127,1000);
45    }
```

FirstMelody: Melodien komponieren durch „cut&paste“

Generieren Sie eine bekannte Melodie, indem Sie Tonhöhen und die Tondauer variieren.

Übung 3:

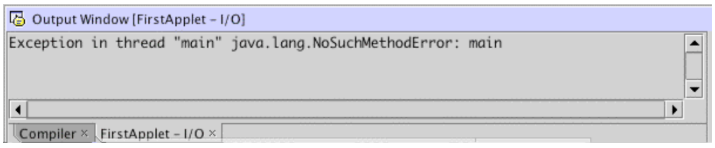
Laden Sie andere Beispielprogramme aus dem CodeKit-Archiv in ihr Verzeichnis. Achten Sie darauf, dass alle notwendigen Klassen und Medien (z.B. Bilder oder Sound-Files) im richtigen Verzeichnis vorhanden sind. Welche Klassen und

Medien-Files von den Java-Applets und Applikationen verwendet werden, ist stets im CodeKit-Archiv (Codebeispiele) mit angegeben. Zum Beispiel: die Applikation „FirstVideo“ benötigt die Klasse „VideoPlayer“ und das Quicktime-Movie „jumps.mov“ und die Erweiterung „qtjava.zip“. Starten Sie die Applikation mit den richtigen Eigenschaften (Properties Execution: external Execution.
Verwendete Dateien:

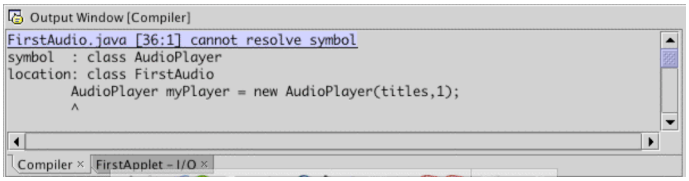
FirstVideo.java
VideoPlayer.java
jumps.mov

TYPISCHE FEHLERMELDUNGEN

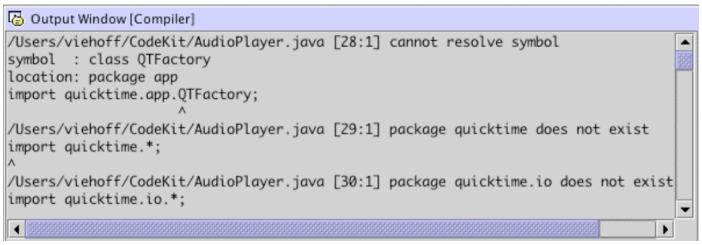
Bei der Übersetzung des Source-Codes („Compilation“) kann eine Vielfalt von unterschiedlichen Fehlermeldungen und Warnungen auftreten, die innerhalb dieser Einführung nicht aufgelistet werden können. Der Programmieranfänger jedoch wird insbesondere bei der ersten Einrichtung seiner Entwicklungsumgebung auf einige typische Fehlermeldungen stoßen, die einfach zu beheben sind.
Einige häufige Java-Fehlermeldungen bei der ersten Nutzung und Erweiterung von CodeKit-Programmen in der NetBeans-Entwicklungsumgebung:



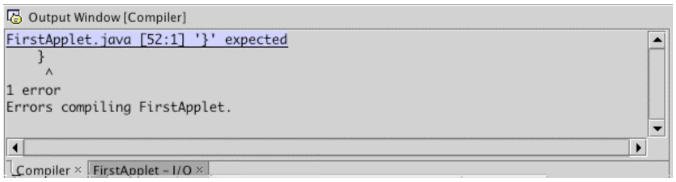
Fehler 1: Ein Java-Applet wird als Applikation ausgeführt. Weil das Applet keine main-Methode hat, wird beim Ausführen des Programms ein Fehler ausgegeben.
Abhilfe: Stellen Sie die Eigenschaften (Properties) auf Execution/AppletExecution.



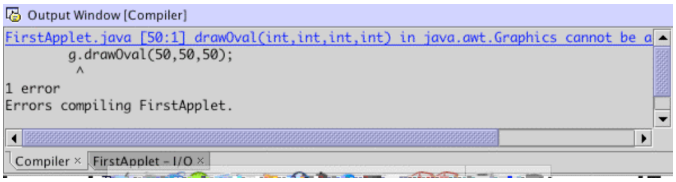
Fehler 2: FirstAudio verwendet ein Objekt der Klasse „AudioPlayer“. Die Klasse AudioPlayer ist nicht im File-System vorhanden, weswegen das Programm nicht compiliert werden kann.
Abhilfe: Speichern Sie AudioPlayer.java in Ihr Arbeitsverzeichnis.



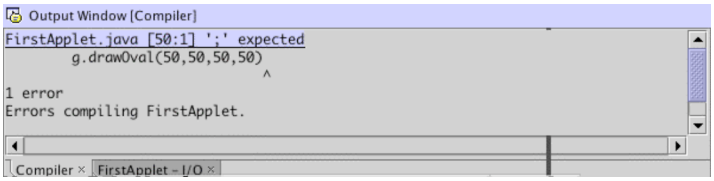
Fehler 3: Die Klasse AudioPlayer verwendet Komponenten von Quicktime for Java, die nicht im File-System vorhanden sind.
Abhilfe: Fügen Sie das Archiv „QTjava.zip“ zum File-System hinzu (File:mount:Archive Files).



Fehler 4: Syntax-Fehler - Es fehlt eine geschweifte Klammer.
Abhilfe: Alle geöffneten Klammerumgebungen müssen auch wieder abgeschlossen werden. Fügen Sie eine geschweifte Klammer hinzu.



Fehler 5: Der verwendete Befehl „drawOval“ hat nicht die richtige Anzahl Parameter
Abhilfe: Ergänzen Sie das fehlende Argument (int) für den entsprechenden Befehl.

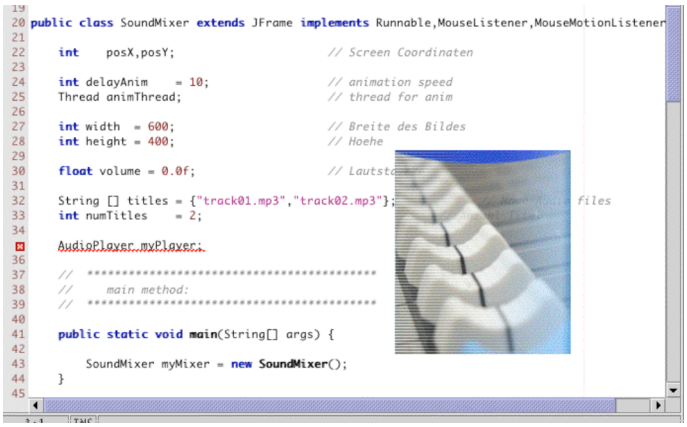


Fehler 6: Die Befehlszeile wird nicht mit einem Semikolon abgeschlossen
Abhilfe: Ergänzen Sie das fehlende Zeichen, um die Zeile abzuschließen.

LINKS

Java 2 Tutorial [http://java.sun.com/docs/books/tutorial/index.html]
NetBeans Guides & FAQs [http://www.netbeans.org/kb/general.html]

CODE & INTERAKTION 1



```
public class SoundMixer extends JFrame implements Runnable,MouseListener,MouseMotionListener

    int    posX,posY;                    // Screen Coordinaten

    int delayAnim    = 10;               // animation speed
    Thread animThread;                   // thread for anim

    int width  = 600;                    // Breite des Bildes
    int height = 400;                    // Hoehe

    float volume = 0.0f;                 // Lautstä

    String [] titles = {"track01.mp3","track02.mp3"};  files
    int numTitles    = 2;

    AudioPlayer myPlayer;

    //  ***************************************
    //     main method:
    //  ***************************************

    public static void main(String[] args) {

        SoundMixer myMixer = new SoundMixer();
    }
```

Die Maus als Mischpult: CodeKit „SoundMixer"

ÜBERSICHT

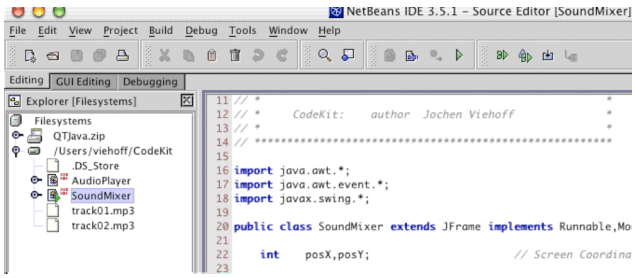
In diesem Abschnitt wird eine interaktive Audio-Applikation aus dem CodeKit-Programmarchiv vorgestellt: der „SoundMixer". Die Applikation spielt zwei Audio-Dateien gleichzeitig mit dem CodeKit-Medienobjekt „AudioPlayer" (basierend auf „Quicktime for Java") ab. Die aktuelle Position der Maus steuert die Wiedergabelautstärken individuell und mischt beide Audio-Dateien zusammen. Verschiedene Übungen verändern oder erweitern die Funktionalitäten des „SoundMixer". Die wichtigen Code-Teile werden kurz erläutert, ohne an dieser Stelle das Programm vollständig erklären zu wollen.

„SOUNDMIXER" HERUNTERLADEN UND STARTEN

Um die Java-Applikation innerhalb der Entwicklungsumgebung NetBeans starten zu können, werden folgende Dateien benötigt:

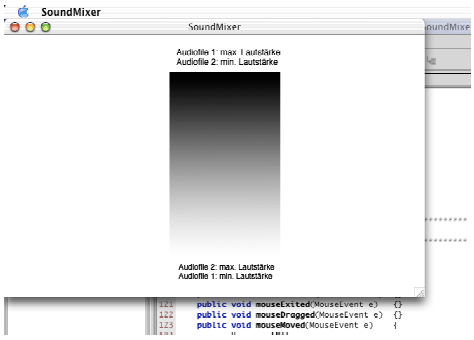
SoundMixer.java
AudioPlayer.java
track01.mp3
track02.mp3

Die Dateien finden sich auch im CodeKit-Archiv unter der Rubrik „AudioControl". Die ausführbare Java-Applikation ist „SoundMixer", die Klasse „AudioPlayer" ist zum Abspielen von mehreren Audio-Dateien mit Quicktime geeignet. „AudioPlayer" selbst ist kein ausführbares Objekt, also kein eigenständiges Programm, das in NetBeans gestartet werden kann. Quicktime unterstützt alle gängigen Audio-Formate (z.B. .wav, .mp3, etc.).
Laden Sie die Dateien in Ihr Arbeitsverzeichnis. Der NetBeans-Explorer sollte alle Files anzeigen:



Ausschnitt: NetBeans-Explorer und Editor

Starten Sie die Applikation „SoundMixer“, mischen Sie beide Musikstücke mit der Vertikalbewegung der Maus über dem Fenster der Applikation:



SoundMixer: Mausposition als Lautstärkeregler

Beide Audio-Dateien werden gleichzeitig und parallel abgespielt. Lediglich die Lautstärken werden entsprechend der Mausbewegung variiert.

Relevante Code-Fragmente

Ohne die Java-Applikation vollständig zeilenweise verstehen zu wollen, sollen an dieser Stelle relevante Abschnitte im Code erklärt werden, sodass Erweiterungen und Abänderungen am Programm vorgenommen werden können - als Ausgangspunkt für eigenes experimentelles Arbeiten.
Um die Audio-Dateien, die sich im gleichen Verzeichnis wie der „SoundMixer“ befinden müssen, laden zu können, müssen die Dateinamen angegeben werden. Dies passiert am Anfang des Source-Codes:

```
27    int width  = 600;                  // Breite des Bildes
28    int height = 400;                  // Hoehe
29
30    float volume = 0.0f;               // Lautstaerke
31
32    String [] titles = {"track01.mp3","track02.mp3"};          // Name Audio files
33    int numTitles    = 2;                                    // Anzahl Titel
34
35    AudioPlayer myPlayer;
36
```

SoundMixer: Liste für die Audio-Dateien

Die Dateinamen werden als „String“ in einer Liste angegeben. An dieser Stelle können Sie den Code ändern, wenn Sie andere Musikstücke im „SoundMixer“ verwenden wollen.
Die Steuerung der Lautstärke beider Audiodateien wird über eine Variable mit dem Namen „volume“ realisiert

```
100
101             // control player - volume
102             if (myPlayer!=null) {
103                 myPlayer.setVolume(0,volume);
104                 myPlayer.setVolume(1,1.0f-volume);
105             }
106         }
```

[SoundMixerCode2.gif]

SoundMixer: Steuerung der Player-Lautstärke

Das Objekt „myPlayer“ ist eine Instanz der Klasse AudioPlayer. Mit dem Befehl „setVolume“ kann sowohl der erste Kanal (myPlayer.setVolume(0,volume);), als auch der zweite Kanal (myPlayer.setVolume(1,volume);) kontinuierlich geregelt werden. Der Wertebereich für die Lautstärke liegt zwischen 0.0 und 1.0. Dass die Lautstärkeregelung entgegengesetzt proportional für beide Kanäle erfolgt, liegt daran, dass Kanal 0 (Track 1) auf den Wert der Variable „volume“ gesetzt wird, Kanal 1 (Track 2) jedoch auf den Wert „1.0f – volume“. Das „f“ kennzeichnet den Varaiblentypen „float“, welcher für den Befehl „setVolume“ vorgeschrieben ist. Dazu ein kleines Rechenbeispiel: Angenommen, der Wert von „volume“ ist „0.8“. Dann ist die Lautstärke von Track 1 „0.8“, die Lautstärke von Track 2 dahingegen „0.2“. Eine Steuervariable regelt beide Lautstärken. Diese Steuervariable „volume“ hängt mit der Mausposition zusammen:

```
112
113     // ****************************************************
114     //   Mouse Events
115     // ****************************************************
116
117     public void mousePressed(MouseEvent e)   {}
118     public void mouseReleased(MouseEvent e)  {}
119     public void mouseClicked(MouseEvent e)   {}
120     public void mouseEntered(MouseEvent e)  {}
121     public void mouseExited(MouseEvent e)   {}
122     public void mouseDragged(MouseEvent e)  {}
123     public void mouseMoved(MouseEvent e)    {
124         posX = e.getX();
125         posY = e.getY();
126         volume = (float)(posY/(double)height);
127
128     }
```

SoundMixer: Maus-Interaktionen

Mit Bewegung der Maus wird in einer Java-Applikation ein entsprechendes „Event“ generiert und die dazugehörige Methode „mouseMoved“ ausgeführt. Die Variablen „posX“ und „posY“ speichern die jeweils aktuelle Mausposition. Damit der Wertebereich von „volume“ zwischen 0.0 und 1.0 liegt, wird der Wert der y-Position („posY“) durch die Breite des Fensters („height“) geteilt. So ist sichergestellt, dass „volume“ keine Werte größer als 1.0 annehmen kann. Rechnen Sie nach! Außerhalb des Fensters werden keine Maus-Events generiert, der größte Wert für „posY“ ist gerade die Höhe des Applikations-Fensters.

ÜBUNGEN

Übung1: Neue Musikstücke

Speichern Sie in Ihr Arbeitsverzeichnis andere Musikstücke in einem von Quicktime unterstützten Format. Ändern Sie den Code im „SoundMixer“ an der richtigen Stelle ab. Achten Sie auf Groß- und Kleinschreibung und auf die richtige Endung („Appendix“).

Übung 2: Maus-Click

Ändern Sie das Programm so ab, dass die Lautstärkeregelung nur bei Maus-Click erfolgt. Hinweis: Bei Maus-Click wird das entsprechene „Event“ erzeugt und die Methode „mouseClicked“ aufgerufen.

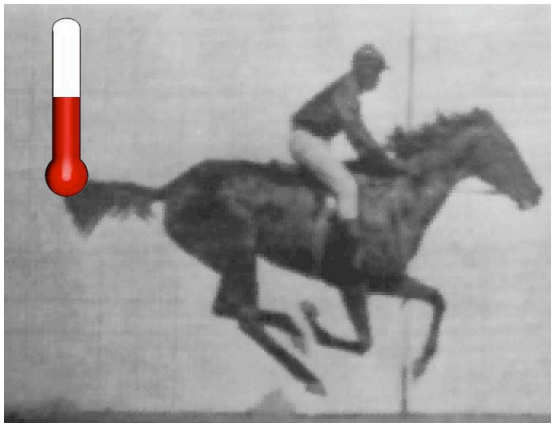
Lösung:

```
112
113     // ******************************************************
114     //   Mouse Events
115     // ******************************************************
116
117     public void mousePressed(MouseEvent e)   {}
118     public void mouseReleased(MouseEvent e)  {}
119     public void mouseClicked(MouseEvent e)   {
120
121         posX = e.getX();
122         posY = e.getY();
123         volume = (float)(posY/(double)height);
124     }
125     public void mouseEntered(MouseEvent e)  {}
126     public void mouseExited(MouseEvent e)   {}
127     public void mouseDragged(MouseEvent e)  {}
128     public void mouseMoved(MouseEvent e)    {}
129 }
130
```

Übung 3: Zweidimensionale Steuerung

Ändern Sie den Programm-Code so ab, dass die Lautstärke des ersten Tracks von der x-Position der Maus und der zweite von der y-Position gesteuert wird. Hinweis: Sie brauchen jetzt zwei Variablen (z.B. „volume1“ und „volume2“), um die Lautstärken getrennt zu regeln.

CODE & INTERAKTION 2



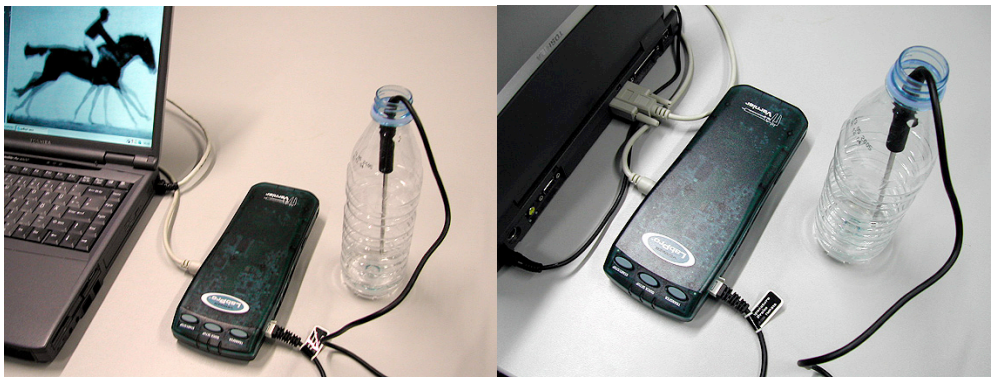
ThermoVideo: Die Temperatur steuert das Video

ÜBERSICHT

Mit dem Vernier LabPro-Meßcomputer [http://www.vernier.com] können unterschiedliche Sensoren an den seriellen Port (nur PC) angeschlossen werden. Die Kommunikation mit einer Java-Applikation erfolgt über das Erweiterungspacket „comm.jar“, die „communication API“ von Sun für Windows-Betriebssysteme. Leider steht für Macintosh-Rechner diese Erweiterung nicht zur Verfügung.
Für die Java-Applikation „ThermoVideo“ wird ein LabPro, ein Temperatur-Sensor und die korrekte Installation der „commAPI“ vorausgesetzt (s. Installation). „ThermoVideo“ misst die Raumtemperatur und steuert mit dem Messwert die Abspielgeschwindigkeit eines Videos. Wird der Sensor erhitzt, läuft das Video schneller. Eingetaucht in Eiswasser kann die Bewegung vollständig „eingefroren“ werden. Die gemessene Temperatur wird als Abspielgeschwindigkeit in den entsprechenden Bereich umgerechnet.

LABPRO-THERMOMETER ANSCHLIEßEN

Verbinden Sie den LabPro-Computer mit dem seriellen Port Ihres PC’s (com1) und schließen Sie den Temperatursensor an „channel 1“ des Messcomputers an:



LabPro Messcomputer und Temperatursensor (Vernier) am seriellen Port (RS232)

Achten Sie darauf, dass die Geräte ausgeschaltet sein sollten, bevor Sie die Stecker verbinden. Der LabPro Messcomputer wurde von der Firma Vernier ursprünglich zur Verwendung im Schulunterricht entwickelt. Eine Vielzahl von Sensoren aus den unterschiedlichsten Bereichen (Physik, Biologie, Chemie) steht zum Experimentieren bereit. Das CodeKit-Programmarchiv bietet Java-Klassen für die meisten LabPro-Sensoren zur Ansteuerung und Auswertung aus einer Applikation heraus (Liste der Sensoren [http://java.khm.de/interface]). Auf diese Weise können Interaktionen für die Ansteuerung von Grafik, Sound oder Bewegtbild genutzt werden.

„ThermoVideo“ herunterladen und starten

Um die Java-Applikation innerhalb der Entwicklungsumgebung NetBeans starten zu können, werden folgende Dateien benötigt:

ThermoVideo.java
Temperature.java
VideoPlayer.java
gallop1.mov

Die Dateien finden sich auch im CodeKit-Archiv unter der Rubrik „VideoControl“. Die ausführbare Java-Applikation ist „ThermoVideo“, die Klassen „VideoPlayer“ und „Temperature“ sind zum Abspielen eines Videos bzw. zur Ansteuerung und Auswertung des Temperatursensors geeignet. „Quicktime“ unterstützt alle gängigen Video-Formate (z.B. .mov, .avi, etc.), nicht jedoch die Komprimierung „mpeg2“. Laden Sie die Dateien in Ihr Arbeitsverzeichnis und installieren Sie die Erweiterungen „qtjava.zip“ und „comm.jar“.

Starten Sie die Applikation „ThermoVideo“. Die Led am LabPro-Messcomputer zeigt den laufenden Messvorgang an, das Video wird gestartet. Testen Sie, ob die Abspielgeschwindigkeit zunimmt, wenn der Sensor erwärmt wird.

Relevante Code-Fragmente

Die Java-Applikation „ThermoVideo“ benutzt zum Abspielen eines Videos das CodeKit-Medienobjekt „VideoPlayer“. Der Dateiname des Videos wird beim „Instanziieren“ des „VideoPlayer“-Objektes angegeben:

```
49          // start video player
50          myPlayer = new VideoPlayer("gallop1.mov",false);
```

ThermoVideo: VideoPlayer starten

Der LabPro-Sensor „Temperature“ wird im Java-Code einfach instanziiert durch:

```
46          // start Thermometer
47          Temperature myThermometer = new Temperature();
```

ThermoVideo: Temperatursensor aktivieren

Es dauert einen Moment, bis der Messcomputer die richtigen Werte liefert (Setup). Eine Schleife wartet, bis die Werte von Null verschieden sind und speichert den ersten „gemessenen“ Wert in der Variable „initTemp“ ab:

```
65        double temp=0.0,tempInit;
66        float  speed = 1.0f;
67
68        // wait till sensor is ready...
69        while (temp==0.0)
70            temp=myThermometer.getTemperature();
71
72        // get init temperature as offset
73        tempInit = temp;
```

ThermoVideo: Initialisierung

Um danach die Temperatur auslesen zu können, benutzt man die Methode „getTemperature()“:

```
79
80            temp = myThermometer.getTemperature();
81
```

ThermoVideo: Auslesen des Sensors

Die Temperatur wird in einer Variable „temp“ gespeichert und in die neue Abspielgeschwindigkeit „speed“ des Videos umgerechnet:

```
81
82            // get new frame rate
83            speed = (float)((temp*temp)/(tempInit*tempInit)-0.5);
84
```

ThermoVideo: Berechnung der neuen Abspielgeschwindigkeit

Beachten Sie, dass die Umrechnung nicht ganz trivial ist. Es wird als Ausgangspunkt die zuerst gemessene Temperatur (beim Start der Applikation: „tempInit“) mitverwendet. Um negative Werte zu vermeiden, wird das Quadrat „temp*temp“ bzw. „tempInit*tempInit“ gebildet.
Nehmen Sie einen Taschenrechner und berechnen Sie ein paar Werte! Welche Geschwindigkeit ergibt sich, wenn die Anfangstemperatur 22.0 Grad und die aktuelle Temperatur 25.6 Grad ist?

Schließlich wird die neue Abspielgeschwindigkeit eingestellt:

```
84
85            myPlayer.setRate(speed);
86        }
```

ThermoVideo: Neue Geschwindigkeit setzen

Die Run-Methode dieser Applikation wird als Loop ausgeführt, d.h. es wird immer wieder eine Temperatur gemessen, einen Moment gewartet, die neue Geschwindigkeit berechnet und schließlich an das „VideoPlayer“-Objekt gesendet. Zur Übersicht, hier noch einmal die gesamte Run-Methode von „Thermo-Video“:

```
59    // ************************************************
60    //  Thread run method
61    // ************************************************
62
63    public void run() {
64
65        double temp=0.0,tempInit;
66        float  speed = 1.0f;
67
68        // wait till sensor is ready...
69        while (temp==0.0)
70            temp=myThermometer.getTemperature();
71
72        // get init temperature as offset
73        tempInit = temp;
74
75        while (runThread != null && runThread.isAlive()) {
76
77            try {Thread.sleep(delay);}
78            catch (InterruptedException e){}
79
80            temp = myThermometer.getTemperature();
81
82            // get new frame rate
83            speed = (float)((temp*temp)/(tempInit*tempInit)-0.5);
84
85            myPlayer.setRate(speed);
86        }
87    }
```

ThermoVideo: Kontinuierliche Messung und Steuerung durch einen Java-Thread

ÜBUNGEN

Übung1: Eigenes Video

Speichern Sie ein eigenes Video im Arbeitsverzeichnis ab und ändern Sie den Source-Code entsprechend ab.

Übung 2: Geschwindigkeitsberechnung

Ändern Sie das Programm „ThermoVideo“ so ab, dass die Abspielgeschwindigkeit zunimmt, wenn die Temperatur fällt (entgegengesetztes Verhalten wie in der Original-Applikation). Wie müssen Sie die Neuberechnung der Variable „speed“ modifizieren?

Übung 3: Zufallsgeschwindigkeit

Diese Übung können Sie insbesondere auch dann verwenden, wenn Sie keinen LabPro-Messcomputer haben. Ersetzen Sie die Temperatursteuerung durch eine Zufallsvariable.
Hinweis 1: Entfernen Sie zuerst alle Komponenten, die etwas mit der Temperaturmessung zu tun haben.

Hinweis 2: Berechnen Sie die Geschwindigkeit des Videos abhängig von einer Zufallsvariablen, die Sie erzeugen können durch:

```
70            // Zufallsvariable im Bereich 0.0 ... 1.0
71            speed = (float)Math.random();
```

Java-Zufallszahlengenerator

Lösung:

```
56    // ************************************************
57    //  Thread run method
58    // ************************************************
59
60    public void run() {
61
62        float  speed = 1.0f;
63
64        while (runThread != null && runThread.isAlive()) {
65
66            try {Thread.sleep(delay);}
67            catch (InterruptedException e){}
68
69
70            // Zufallsvariable im Bereich 0.0 ... 1.0
71            speed = (float)Math.random();
72
73            myPlayer.setRate(speed);
74        }
75    }
```

Übung 4: Experts only!

Ersetzen Sie die Temperatursteuerung in „ThermoVideo“ durch eine Maus-Interaktion, indem Sie die passenden Methoden und Befehle aus der Java-Applikation „SoundMixer“ einfügen.