Techischer Report: TRecS : Ein tangibles, rekonfigurierbares System zur explorativen Datenanalyse

Jean René Dawin, Danny Hartwig,
Andreas Kudenko, Eckard Riedenklau

Universität Bielefeld, Sommersemester 2007

Inhaltsverzeichnis

1 Motivation

Durch die Virtualisierung von Information verlieren Menschen das Gefühl für die Beschaffenheit dieser digitalen Information. Z.B. wird der Festplattenspeicher lediglich in einer Zahl ausgedrückt. Da sich die Maße der Festplatte nicht ändern, wenn Daten auf ihr gespeichert werden, kann man nur anhand dieser virtuellen Zahl den freien Speicherplatz ermitteln. Digitale Informationen sind folglich nicht anfassbar und werden durch das Verlieren physischer Attribute durch Digitalisierung immer abstrakter und unverständlicher. Ein weiteres Beispiel sei Geld. Viele Menschen zahlen bevorzugt mit Kreditkarten, dem sog. „Plastikgeld“. Sie kaufen damit ein und merken u.U. gar nicht, dass sie immer mehr Schulden anhäufen. Durch die Nutzung von realem Papier- und Münzgeld hat man ein besseres Verständnis für den eigentlichen Wert des Geldes – auch und gerade durch das Anfassen.

Dinge anzufassen, um sie zu begreifen, d.h. zu verstehen, ist sehr wichtig für ein tiefes Verständnis der Dinge. Durch das 'Begreifbar'-machen von Information (oder zumindest das Erfahren und Erkunden) wird eine reale Verbindung zwischen Mensch und Information hergestellt. So ist es durch Anfassen und Manipulieren möglich, etwas über die Beschaffenheit der Dinge zu erfahren. Dies ist die Quintessenz von Tangible Computing.

Tangible Computing versucht zwischen der realen, physischen und der virtuellen, digitalen Welt zu vermitteln und die Vorteile beider zu nutzen, um Informationen besser verständlich zu machen.

Wegen der intuitiven Benutzbarkeit, die bei der Nutzung von Tangible Computing entsteht, wurde es bereits in vielen Anwendungsgebieten und Domänen genutzt. Beispiele sind klangverarbeitende Systeme (ein sehr prominentes Gebiet für Tangible Computing; REACtable [JKGB05] und AudioPad [PRI02] sind nur wenige Beispiele), Datenaustausch [UIG98], Arbeitsplatzerweiterungen oder Alltagsinterfaces, wie z.B. die sog. „Marble Answering Machine“ (Murmel-Anrufbeantworter) [IU97].

In allen uns bekannten Implementationen wird Tangible Computing für eine bestimmte Aufgabe und zumindest ein bestimmtes Anwendungsgebiet genutzt. In dieser Arbeit entwickelten wir eine neuartige Strategie, um Tangible Computing für ein möglichst weites Feld von Anwendungsgebieten nutzbar zu machen.

Als Domäne für unser System haben wir tabellarische Datensätze gewählt. Normalerweise werden Daten in einer Datenbank oder das Resultat einer Anfrage an eine relationale Datenbank in tabellarischer Form organisiert. Sogar Zeitserien werden tabellarisch organisiert, indem man pro Zeile einen diskreten Zeitpunkt abbildet.

Ganz allgemein können auf diese Weise sehr hochdimensionale, multivariate Daten organisiert und verarbeitet werden. Also kann unser System mit (fast) allen Arten von Daten umgehen. Natürlich gibt es auch Domänen, die in diese Repräsentation nicht eingepasst werden können. Die Domäne *Text* ist ein Beispiel hierfür. Schriftstücke lassen sich nicht in tabellarische Form pressen, ohne Informationen zu verlieren oder zu verändern.

Zielsetzung

Alle oben vorgestellten tangiblen Systeme sind nicht universell einsetzbar, d.h. sie dienen jeweils nur einem ganz bestimmten Zweck. Zielsetzung dieses studentischen Projektes war es daher, ein neuartiges (single-user) System zu entwickeln, das dem Benutzer eine nicht-spezialisierte Tangible Arbeitsumgebung bietet.

Bildlich gesprochen sollte ein Werkzeugkasten mit unterschiedlichen Werkzeugen (Tools) und eine Sammlung von verschiedenen Materialien (Datensätze) entstehen. Dem Nutzer sollte es frei stehen, wie er die Tools und Datensätze kombiniert, um aus diesen neue Informationen und Einsichten zu gewinnen. Der Nutzer selbst bestimmt die Funktionen des entstehenden Systems. So gestaltet er ein Werkzeug zur explorativen Datenanalyse nach seinen Vorstellungen.

Anwendung

Das System soll dem Benutzer ein Framework bieten, in dem er die Möglichkeit hat, abstrakte Daten audiovisuell erfahrbar zu machen. Dabei soll das System den Nutzer sowohl visuell, als auch akustisch in den Daten navigieren lassen können. Es soll möglichst beliebig viele Daten und Funktionalitäten unterstützen. Lediglich das implementatorische Geschick des Anwenders / Programmierers setzt hier Grenzen.

2 Theoretischer Hintergrund

TRecS (Tangible Reconfigurable System) ist ein single-user System, das dem Benutzer ermöglicht, explorative Datenanalyse zu betreiben. Hierzu werden Objekte visuell verfolgt und virtuell mit Funktionen und Daten verknüpft. Diese kann der Benutzer beliebig mit Hilfe der Objekte in Interaktion bringen.

Es werden in TRecS verschiedene Techniken und Konzepte angewandt, die wir in diesem Kapitel erläutern. Zuerst gehen wir auf das Forschungsgebiet des Datamining und der explorativen Datenanalyse ein. Anschliessend beschreiben wir, was unter der Sonifikation zu verstehen ist. Abschnitt 2.3 behandelt das zentrale Interaktions-Konzept des Tangible Computings. Projekte, die aus dem gleichen Umfeld kommen und unserem Projekt ähnlich sind, beschreiben wir in Abschnitt 2.4.

2.1 Datamining und Explorative Datenanalyse

Unter Datamining fasst man Techniken zusammen, die es ermöglichen, in großen und unübersichtlichen Datenmengen Strukturen und Muster zu finden. Es dient somit der Wissens- und Erkenntisgewinnung in Daten.

Dabei werden in diesem weiten Forschungsfeld statistische Verfahren, Methoden der Dimensionsreduktion (PCA, SVD, Clustering, etc.), Visualisierung (z.B. Scatter-Plots), Klassifikation und Klassifikationsbäume und Verfahren aus den verwandten Gebieten Machine Learning, Mustererkennung und Neuro Computing genutzt.

Ein weiteres Teilgebiet des Dataminings ist die explorative Datenanalyse. Bisher nutzen wir hieraus lediglich die Visualisierungstechnik „Scatter-Plot“. Ein Beispiel für eine Scatter-Plot-Matrix liefert Abb. 12. Dabei werden einzelne Dimensionen eines Datensatzes gegeneinander in einem Plot aufgetragen. So lassen sich z.B. lineare Abhängigkeiten zwischen einzelnen Dimensionen oder andere interessante Muster erkennen.

2.2 Sonifikation

Sonifikation meint die Verklanglichung von Daten und lässt sich allgemein wie folgt definieren:

> *Sonifikation ist die Nutzung von nichtsprachlichen Klängen, um Informationen auszudrücken. Genauer gesagt ist Sonifikation die Transformation von Datenzusammenhängen in wahrnehmbare Zusammenhänge in einem akustischen Signal, um sie zu übermitteln und zu interpretieren. [Her]*

Da Musiker und Klang-Künstler begonnen haben, ihre Kompositionen Sonifikation zu nennen, Sonifikation aber als wissenschaftliche Datenexplorationsmethode und nicht als Kunst verstanden werden soll, wurde eine präzisere Definition formuliert:

> *Jegliche Technologie, die Daten als Eingabe nutzt und Klang-Signale (evtl. als Antwort auf Anregung) generiert, kann Sonifikation gennannt werden, wenn und nur wenn*
>
> 1. *der Klang Eigenschaften und Zusammenhänge der Daten widerspiegelt.*
> 2. *die Transformation systematisch erfolgt. D.h., dass es eine präzise Definition für das Zusammenspiel von Daten, Interaktion und Klang-Änderung gibt.*
> 3. *die Sonifikation reproduzierbar ist: Dieselben Daten und identische Interaktion resultieren in strukturell identischen Klängen.*
> 4. *die Sonifikation absichtlich mit verschiedenen Daten genutzt werden kann, aber auch wiederholt mit den selben. [Her08]*

Bisher wurden folgende Sonifikationstechiken beschrieben:

Audifikation Bei der Audifikation werden die Datenwerte als instantane Schalldruckwerte interpretiert. Die Zahlen werden direkt in ein Signal überführt.

Auditory Icons Auditory Icons sind schnell verstehbare Alltagsgeräusche und markieren Ereignisse (Beispiel: Papierknistern bein Löschen von Dateien).

Earcons Earcons sind kurze Motive (Melodien und Rhythmen), die genutzt werden, um sog. Verbund-Nachrichten zu repräsentieren (Beispiel: Ein erfolgreiches Aufbauen einer Netzwerkverbindung wird mit einer kurzen Tonfolge ansteigender Höhe kommuniziert).

Parameter Mapping Sonifikation Bei der Parameter-Mapping Sonifikation wirken sich Datenänderungen direkt auf die Klangattribute (Tonhöhe, Laufstärke, Klangschärfe, etc.) aus und beeinflussen dadurch das Klangbild.

Model-based Sonifikation Hier werden die Daten zur Konstruktion eines dynamischen Modells genutzt, das der Nutzer geziehlt anregen, also spielen kann.

In TRecS machen wir beim Tool „DataScanning“ [BHR06b] Gebrauch von modellbasierter Sonifikation. DataScanning ist ein exploratives Datenanalyse-Werkzeug, das es dem Nutzer erlaubt, hoch-dimensionale Datenverteilungen mittels eines physischen Objektes zu untersuchen. Dieses dient zur interaktiven Steuerung der verwendeten modellbasierten Sonifikation.

2.3 Tangible Computing

TRecS kommt nicht ohne zentrale Konzepte des Tangible Computings aus. Die von uns genutzten Konzepte werden an dieser Stelle angesprochen.

Tangible Computing ist ein Teilgebiet des Human-Computer-Interfaces (HCI) Designs. Es werden hier neue Interaktionsmöglichkeiten zwischen Mensch und Maschine erprobt, die die Eigenschaften von dinghaften und begreifbaren, also physischen Objekten ausnutzen.

Physikalische vs. virtuelle Repräsentation

Im Tangible Computing wird (speziell bei Tisch-Applikationen, wie auch TRecS eine ist) eine Verbindung aus physikalischer und virtueller Repräsentation für Daten und / oder Funktionalität genutzt. Die physikalische Repräsentation bildet die für den Menschen begreifbare Komponente der Mensch-Maschine-Interaktion und die virtuelle das Gegenstück im Weltmodell des Computers. Die Verbindung beider Repräsentationen führt bei geschicktem Design zu einer Kombination der Vorteile beider Repräsentationen.

Der Nutzer ist es im Alltagsleben gewöhnt, mit physischen Objekten zu hantieren und kann dies auch intuitiv. Z.B. ist aber die Nutzung einer Computer-Maus ist hiervon zu differenzieren. Sie ist zwar physikalisch, dient aber nur der Kontrolle eines virtuellen eines Mauszeigers auf einem Bildschirm. Die Maus ist also nicht der Mauszeiger. Die Fähigkeit zum Nutzen einer Maus muss erst erlernt werden und ist weniger intuitiv, als das direkte Manipulieren realer Objekte.

Building while Playing

Des Weiteren führen wir den Begriff „Building while Playing“ ein. Dieser meint, dass das Konfigurieren des Systems mit dem Benutzen gleichzusetzen ist. D.h. durch Umkonfigurieren des Systems wird die Arbeitsweise geändert. Umgekehrt wird aber auch durch das Arbeiten am System kontinuierlich die Konfiguration verändert.

Container-Konzept

Das Container-Konzept wurde in einer tangiblen Umgebung zuerst in dem Projekt mediaBlocks [UIG98] genutzt. Dieses Konzept besagt, dass mit zunächst bedeutungslosen Objekten Informationen verknüpft werden können, um ihnen eine Bedeutung zuzuschreiben. Die Information kann von ganz unterschiedlicher Natur sein.

Im mediaBlocks Projekt waren es z.B. Bilder und Videos, es könnten aber auch Text, oder wie im TRecS abstrakte Datensätze und Funktionalität sein (s. Abschnitt 3.1).

Bedeutung und Kopplung

In [Dou04] wird festgestellt, dass Menschen den Objekten eine Bedeutung geben. Desweiteren stellt der Autor fest, dass Objekte bei einer Interaktion „gekoppelt“ werden.

Darauf aufbauend weist der Autor auf zwei Prinzipien hin:

1. *Users, not designers, create and communicate meaning.*
2. *Users, not designers, manage coupling.*

Das erste Prinzip besagt, dass Benutzer Objekten Bedeutung geben und sie mit anderen Benutzern austauschen, der Designer aber darauf keinen Einfluss hat. Genau so steht es mit der Kopplung im zweiten Prinzip: Auch hier entscheidet nicht der Designer, sondern der Benutzer, wie er Objekte koppelt, d.h. in Interaktion bringt. Dabei können völlig neue Möglichkeiten von Interaktionen zwischen den Objekten entstehen. Wir nennen dies *emergente Funktionalität.*

Diese Erkenntnisse sind sehr wichtig beim Design von interaktiven Systemen, wie TRecS. In TRecS sollten diese postulierten Prinzipien belegt werden. Dadurch, dass man dem Benutzer überlässt, wie er die Objekte mit Funktionalität und Daten (meaning) belegt und wie er sie in Interaktion bringt (coupling), soll emergente Funktionalität entstehen.

Zusammenhänge

In TRecS wird versucht, alle diese angesprochenen Techniken und Konzepte in TRecS zu vereinen, um somit ein nicht-spezialisiertes, tangibles und rekonfigurierbares System zu entwickeln. Dabei werden die Konzepte des Tangible Computings in einem zentralen Interface genutzt, um dem Nutzer die Methoden und Techniken aus den Gebieten Datamining und Sonifikation quasi „an die Hand“ zu geben.

2.4 Verwandte Projekte

Es gibt technisch oder konzeptionell verwandte Projekte zu TRecS . Diese Systeme verfolgen aber alle spezifische Anwendungsfälle. Bei fast allen handelt es sich um Tisch-Applikationen, d.h. die Interaktion zwischen Nutzer und Computer findet auf einer interaktiven Tischoberfläche statt, auf der mit Hilfe von Tracking-Systemen Objekte verfolgt werden. Lediglich mediaBlocks ist keine reine Tisch-Applikation.

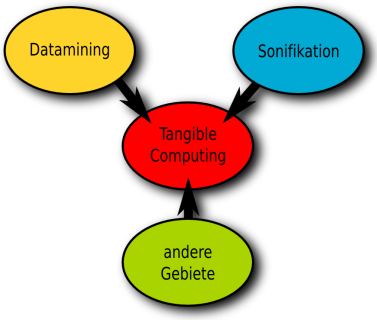


Abbildung 1: Die genutzten Forschungsgebiete und wie sie zusammenhängen

metaDESK

Abbildung 2: metaDESK

metaDESK ist eine Plattform für das Erforschen des Designs von tangiblen Benutzerschnittstellen. Es integriert mehrere 2D und 3D Ansichten mit einer Zusammenstellung von Gegenstände und Instrumente, die mit eine Reihe von optischen, mechanischen und elektromagnetischen Sensoren erfasst und verfolgt werden. So angereichert mit virtueller Funktionalität, verknüpft metaDESK diese Objekte als fühlbare Schnittstellen mit einer Reihe graphisch-intensiver Anwendungen.

Mit der metaDESK Plattform werden unterschiedliche Aspekte untersucht:

1. die Verkörperung von GUI-Elementen, wie Symbolen, Widgets und Fenstern
2. die Koppelung täglicher Gegenstände mit numerischer Information, die mit ihnen verknüpft wurden.

Tangible Geospace ist eine erste Anwendung der metaDESK Platform. Tangible Geospace nutzt physische Modelle von diversen Landmarken als sog. Phicons (physische Icons), um dem Benutzer die Möglichkeit zu geben, die 2D- und 3D-Ansichten von graphischen Karten zu manipulieren.

Gleichzeitig dient eine an einem beweglichen Arm montierte „active LENS“ als Display für eine räumlich kontinuierliche 3D-Ansicht. Durch Greifen und Bewegen der active LENS (ein physisch verkörpertes Fenster), kann der Benutzer in der 3D-Asicht navigieren. Tangible Geospace führt außerdem das Konzept der „passive LENS“, einer passiven optischen Ansicht ein, die als unabhängiges Display auf der Projektionsfläche des metaDESK fungiert. [UI97]

mediaBlocks

Abbildung 3: mediaBlocks

Im Projekt *mediaBlocks* [UIG98] werden Objekte, die über RFID-Tags verfügen, an Datenquellen, die mit RFID-Tag-Readern (transfer slots) ausgestattet sind, mit Daten gefüllt. Als Datenquellen dienen dabei z.B. eine Videokamera oder ein digitales Whiteboard. Mit den Daten der gefüllten mediaBlocks kann mit Hilfe von Manipulatoren (speziellen tangiblen Interfaces, mit der sie betrachtet und neu arrangiert werden können) gearbeitet werden. Ist der Benutzer mit dem Ergebnis zufrieden, kann er die mediaBlocks wiederum in transfer slots von Displays einlegen, um die Daten z.B. auf einem Drucker auszugeben.

Urp

Urp ist ein System zur Städteplanung. Es integriert eine breite Palette von Funktionen zur Planung von Städten in eine einzelne physische Arbeitsumgebung. Die I/O Bulb Infrastruktur, auf der die Applikation basiert, erlaubt physische Architekturmodelle auf eine gewöhnliche Tischoberfläche zu platzieren, die sogar tageszeit-abhängigen Schattenwurf, bei Glasoberflächen Lichtreflektionen, echt-zeit Windsimulation, etc. ermöglichen. [UI99]

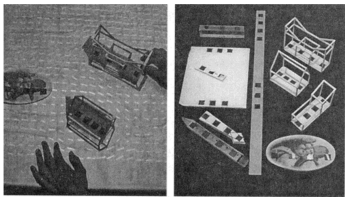


Abbildung 4: Urp

AudioPad

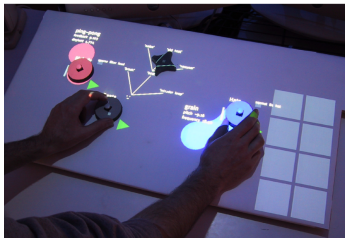


Abbildung 5: AudioPad

AudioPad [PRI02] ist ein kollaboratives sample-, bzw. loopbasiertes Liveperformance-Instrument. Es nutzt RF-Tracking, um die Position und eindeutige ID der Objekte auf der Oberfläche zu bestimmen. Um auch die Rotation eines Objektes bestimmen zu können, wird ihm ein zweiter, leicht versetzter Tag hinzugefügt. Der Schaltkreis (prinzipiell Spule und Kondensator) des zweiten Tags kann vom Benutzter mittels eines Tasters unterbrochen werden.

Das Bedienkonzept ist wie in TRecS stark container- und controllerorientiert. D.h., dass die an sich gleichförmigen Objekte auf „Lade- und Entladeflächen“ mit Samplebänken beladen werden können (Container-Konzept). Auf dem übrigen großen Bereich der Interaktionsfläche kann mit Hilfe eines Auswahlobjektes (sternförmig) das baumartige Samplemenü der einzelnen beladenen Objekte durchsucht (Controller-Konzept) und Samples ausgewählt werden. Auf ähnliche Weise

werden auch die Parameter mithilfe des Auswahlobjektes beeinflusst. Des Weiteren gibt es auch Mikrophon- und Filterobjekte.

reacTable

Abbildung 6: reacTable

reacTable [JKGB05] ist als modularer Synthesizer gedacht, dessen Klanggeneratoren und Kontrollmodule durch Objekte repräsentiert werden. Das Trackingsystem ist visionbasiert und nutzt, genau wie TRecS , Fiducial-Marker zur Erkennung und Verfolgung der Objekte.

Gedacht ist der reacTable vornehmlich zum Sound-Design, aber auch zum Komponieren und kann kollaborativ von mehreren Personen genutzt werden. Hinzu kommt, dass mehrere reacTables (sogar über das Internet und damit weltweit) vernetzt werden können, was verteilte Live-Performances ermöglicht. Eine physische Repräsentation auf dem einen reacTable wird auf den (anderen) reacTables nur rein virtuell repräsentiert, da reale Objekte z.Z. nicht nachgeführt werden können. Nachteil daran ist natürlich, dass nur derjenige Nutzer dieses Objekt direkt beeinflussen kann, der Zugriff auf die physische Repräsentation hat.

Der Aufbau eines Synthesizer-Konstruktes ist äquivalent mit dem Spielen des Synthesizers. Wird eine neue Note eingestellt oder ein Parameter eines Klanges verändert, so ändert der Musiker den Aufbau des Synthesizers. Parameter wie die Frequenz werden z.B. durch Rotation des jew. Objektes, die Amplitude durch Umfahren des Objektes mit einer Fingerspitze geändert. Diese muss mit einem speziellen Fiducial-Marker versehen sein, damit das System diese wahrnehmen kann. Beim Aufbau / Spielen des Synthesizer sei noch zu erwähnen, dass auch redundante Aufbauten möglich sind.

Die zugrundeliegende Klangerzeugungsengine (SuperCollider, bzw. PureData) – angesteuert über Open Sound Control (OSC) – wird durch die Kombination verschiedenförmiger Objekte beeinflusst, die auch unterschiedliche Einflüsse auf den Synthesizer-Aufbau haben. Sie sind durch unterschiedliche Formen repräsentiert, z.B. Oszillatoren (flache Quader), Metronome (Halbkugeln) oder auch Filter.

music table

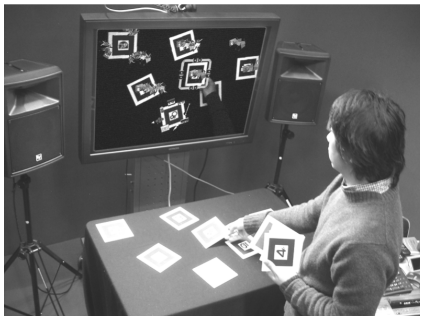


Abbildung 7: musicTable

music table [BMHS03] ist ein Sequencing-Interface, das durch sein Setup nur bedingt kollaborativ genutzt werden kann. Es nutzt einen Augmented Reality-Ansatz, basierend auf dem ARToolkit [art08], welches es ermöglicht, mit einer Kamera die Lage von Markern mit ihren 6 DOF im dreidimensionalen Raum zu verfolgen.

Mit diesen Markern ausgestattete Karten erhalten auf der Tischoberfläche des music table durch Augmented Reality Bedeutung. Diese Karten werden auf einem Display, auf dem die beobachtete Tisch-Szene dargestellt wird, mit Darstellungen von virtuellen Kreaturen und Steuerungskonstrukten angereichert. Diese virtuellen Kreaturen repräsentieren die zu spielenden Noten, bzw. Sequenzen. Diese können durch Container-Karten, auf denen mehrere Kreaturen „leben“, zusammengefasst werden. Instrumentfarben werden mit Hilfe von kippbaren Kontrollkarten ausgewählt.

Problematisch an diesem Augmented Reality-Ansatz ist die Tatsache, dass der Benutzer permanent auf ein Display schauen muss, um sowohl das reale Geschehen auf dem Tisch, als auch die virtuellen Repräsentationen im Blick zu haben. Hierfür wäre ein Setup denkbar, das eine VR-Brille mit zusätzlicher Kamera nutzt.

Tasting Music

Das Tasting Music System [Miz] dient zur Abfrage von Musik aus einer Datenbank. Bei der Entstehung von Tasting Music sind die Entwickler ganz pragmatisch vorgegangen. Sie haben sich die Umgangs- und Bedienkonzepte von Audio CDs und CD-Playern angesehen, um diese durch tangible Ansätze zu verbessern. Dabei sind sie beim bekannten Konzept der CD(-Hülle) geblieben und nutzen diese als hartes Container-Konzept für Alben und Sampler, wie man es schon immer bei CDs gewohnt ist.

Abbildung 8: Tasting Music

Um die Funktionsweise zu erklären, soll hier ein kurzer Usecase skizziert werden: Der Benutzer sucht sich aus seiner CD-Sammlung ein Exemplar aus, welches er sich anhören möchte. Statt zum CD-Player zu gehen und dort die CD aus der Hülle zu nehmen und in den Player einzulegen, legt er die Hülle (egal, ob mit oder ohne CD) auf eine sensitive Tischoberfläche, die erkennt, um welche CD es sich handelt. Da alle CDs inklusive der Audio-Daten in der Datenbank des Tisches gespeichert sind, wird daraufhin auf dem Tisch ein Tracklisting der jew. CD projiziert, aus dem der Benutzer mit einem tangiblen Objekt (in Form eines Pucks) einen gewünschten Track auswählen und abspielen kann.

3 Konzeption

3.1 Das System

Im Projekt ist mit TRecS ein modulares und einfach zu erweiterndes interaktives System entstanden.

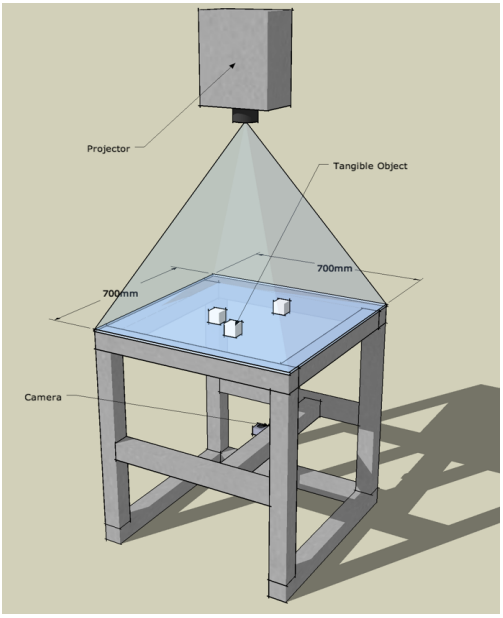


Abbildung 9: tDesk-Hardware

Aufgebaut ist dieses System als Tisch-Applikation. Hierzu wurde der tDesk[HHR04] als Hardware-Grundlage verwendet.

Das Hardware-Setup (siehe Abb. 9) wurde im Gegensatz zu vorangegangenen Entwicklungen leicht verändert. Über der Tischoberfläche wurde ein Projektor angebracht über den es möglich ist Visualisierungen und graphische Elemente des TUIs (Tangible User Interface) zu projizieren. Um die mit Fiducial-Markern [BKJ05] versehenen Objekte auf dem Tisch zu erkennnen und zu verfolgen wurde unter der Glasplatte des Tisches eine Kamera platziert. Auf der Glasplatte liegt eine Diffusor-Folie, sodass nur Objekte die direkt auf dieser Folie stehen, von der Kamera scharf erfasst werden. Softwareseitig wurden folgende Komponenten verwendet:

TUIO-Protokoll Über das UDP basierte TUIO-Protokoll [KBBC05] werden die Marker-Daten von der Tracking-Software über ein Netzwerk verschickt.

SETO Die Software SETO [Bov07] erhält die Tracking-Informationen und verwaltet die Objekte.

SuperCollider Implementiert ist das gesamte System inklusive der oben genannten Softwarekomponenten in SuperCollider mit SwingOSC-Unterstützung [swi07]. SwingOSC wurde zum rendern der TUI und zur Visualisierung der Datensätze auf der Tischoberfläche genutzt.

Die interaktive Tischoberfläche wurde in zwei horizontal getrennte Bereiche eingeteilt (s. Abb. 10).

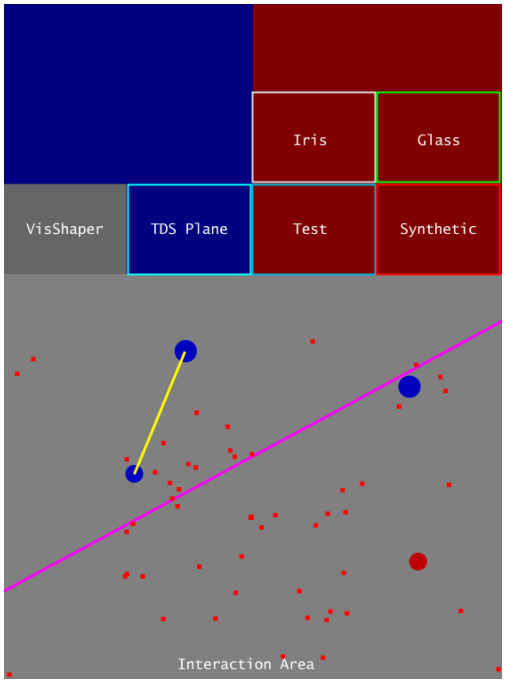


Abbildung 10: Layout der Tischoberfläche. Die großen Punkte stellen virtuelle Repräsentationen von Objekten dar.

Im oberen Bereich können die physischen Objekte mit virtuellen Bedeutungen bzw. States versehen werden. Im unteren Bereich findet dann die eigentliche Interaktion der Objekte statt. Dabei erzeugt SETO sogenannte Interactions, in denen nun auf einen konkreten Datensatz eine Funktionalität angewandt wird. Dabei entsteht ein geschlossener Interaktionskreislauf zwischen Benutzer und System.

Allgemein wird zwischen zwei Arten von Objekten unterschieden: den Datenobjekten und den Toolobjekten. Datenobjekte halten und verwalten allgemeine tabellarische Datensätze (jeweils einen pro Objekt). Des weiteren gibt es auch die Möglichkeit, die Daten, die die Objekte halten, durch den zugehörigen virtuellen State zu visualisieren. Toolobjekte hingegen beinhalten Funktionen, die auf Daten arbeiten. Auch die Toolobjekte können Informationen, wie die Interaktionen die zwischen zwei Objekten stattfinden, zeichnen.

Die Software ist so ausgelegt, dass die Funktionalitäten gleichzeitig und damit auch nebenläufig arbeiten. Paarweise Interaktion zwischen zwei Objekten ist generell erlaubt. Jede weitere Funktionalität erhöht die Anzahl möglicher Interaktionen, sofern diese nicht explizit ausgeschlossen werden. Dadurch kann der Nutzer das System auch in einer Weise benutzen, die beim Entwurf des Systems nicht bedacht wurde. Der Benutzer entscheidet also mit, wie das System zu benutzen ist und welche Funktionalitäten sinnvoll sind.

Die Interaktion des Benutzers mit dem System erfolgt durch Veränderung der Objektkonfiguration auf der Tischoberfläche. Das System erkennt und verarbeitet diese Interaktion und reagiert darauf durch Anpassung der Visualisierung oder durch Sonifikation. Es entsteht eine geschlossene Interaktionsschleife (s. Abb.: 11). Je kürzer die Zeit zwischen Nutzer-Interaktion und System-Antwort ist, desto direkter ist das Arbeiten mit dem System. Im Idealfall ist zwischen Interaktion und Antwort gar keine Latenz, sodass das System in Echtzeit arbeitet.

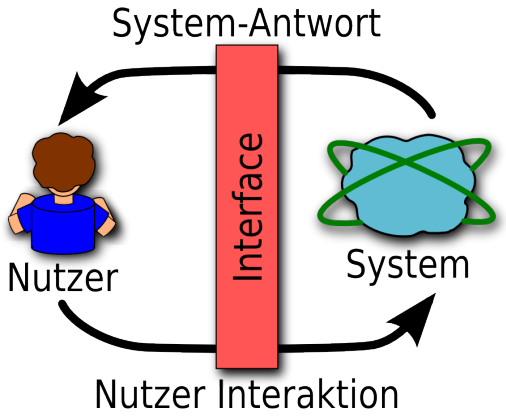


Abbildung 11: geschlossene Interaktionsschleife

3.2 Die Datensätze

Jedes Objekt kann mit einem Datensatz assoziiert werden. Die Datensätze stellen bei TRecS die wichtigsten Komponenten dar. Schliesslich möchte der Benutzer des Systems diese erforschen und verstehen. Dem Benutzer ist dabei selbst überlassen, welche Datensätze er auswählt und wieviele. Es besteht die Möglichkeit, mehrere Datensätze gleichzeitig zu visualisieren und sie mit den bereitgestellten Tools zu untersuchen. Dabei haben wir uns für folgende Datensätze entschieden, die dem Benutzer zur Auswahl stehen:

Iris ist ein vierdimensionaler Datensatz, dessen Datenpunkte beim Vermessen von Blüten gewonnen wurden [iri88]. Zur Verwendung im TRecS wurde auf die vierte und fünfte Dimension verzichtet und falls nötig instantan eine 2D-Projektion

erstellt.

Glass ist ein Datensatz mit neun Attributen. Diese geben die chemischen Zusammensetzungen verschiedener Glassorten an. Er enthält ingesamt sieben Glassorten. [gla87]

Synthetic ist ein dreidimensionaler, von uns synthetisch erzeugter Datensatz. Er enthält zufällig verteilte Datenpunkte.

Test enthält drei Datenpunkte und ist nur für die Entwicklung und Erprobung der Tools von Bedeutung.

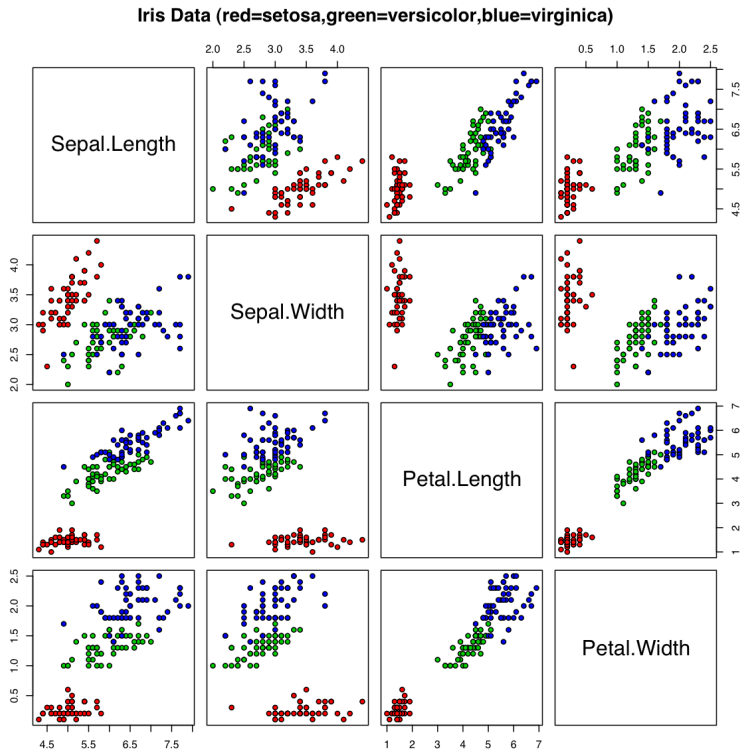


Abbildung 12: Scatterplot-Matrix des Iris Datensatzes [iri08]

Die Anzahl, der zur Auswahl stehenden Datensätze, ist nicht beschränkt. TRecS ist so konzipiert, dass eine Erweiterung möglich und erwünscht ist.

3.3 Die Tools

Jedes Objekt kann mit jeweils einem Tool, der eigentlichen Funktionalität des Systems, assoziiert werden. Die bisher implementierten Tools entstammen den Gebieten des Dataminings und der Sonifikation. Diese sind

- VisShaper
  ändert die Visualisierung eines Datensatzes (ähnlich der Multitouch-Interaktionen [Han]). Entsprechend der Bewegung der VisShaper-Objekte werden die Datenpunkte rotiert, gezoomt oder verschoben.

- Tangible Data Scanning (Model-based Sonification) [BHR06a]
  ist eine virtuelle Ebene, die durch Datenpunkte "bürstet" und durchquerte Punkte zum Klingen anregt. Jeder Datenpunkt "hängt" sozusagen an einer Feder und wird beim Durchqueren der Ebene in Schwingung versetzt, die der Nutzer hören kann.

Manche Tools lassen sich mit mehr als einer Instanz des jeweiligen Tools nutzen. So erhält z.B. das VisShaper-Tool die Funktionalität, die Visualisierung auch zu zoomen. Auch andere Tools und Datensätze lassen sich mehrfach in der Interaktionsfläche nutzen.

Die wenigen bisher implementierten Tools dienen lediglich der Veranschaulichung des Systems. Das System ist modular konzipiert, sodass es sehr einfach um weitere Tools erweitert werden kann. Es lassen sich natürlich auch Tools aus anderen Bereichen als denen des Datamining und der Sonifikation implementieren.

4 Umsetzung

4.1 System

Wie schon eingangs erwähnt fußt das System hardwareseitig auf dem bereits existenten tDesk-System (siehe Abb. 9). Als Framework wird SETO [Bov07] genutzt, welches in SuperCollider [sc07] implementiert wurde.

Wie in Abb. 13 zu sehen, werden die SETO-Klassen `TUIObject` und `TUIOInteraction` für TRecS zu `TRecSObject` (s. Listing 1) und `TRecSInteraction` (s. Listing 3) abgeleitet. Sie wurden u.a. um das Attribut `vState` und die Funktionen `transit()` erweitert. Hinzu kommen noch weitere Attribute und Funktionen, die zur internen Verwaltung nötig sind.

Das Attribut `vState` hält den jew. aktuellen Zustand des Objekts / der Interaktion. Diese Zustände sind alle von den Klassen `TRecSObjectState` (s. Listing 2) bzw. `TRecSInteractionState` (s. Listing 4) abgeleitet. Diese Klassen enthalten letztendlich die einzelnen Daten und Funktionalitäten des Systems und sind wie schon erwähnt hierarchisch organisiert (s. Abb. 14 und 15).

Die Funktion `transit()` entscheidet basierend auf dem Zustand der Objekte, d.h. der physischen Position und dem aktuellen virtuellen Zustand, ob und welcher neue virtuelle Zustand dem `TRecSObject` zugewiesen wird. Analog wird bei der `TRecSInteraction`

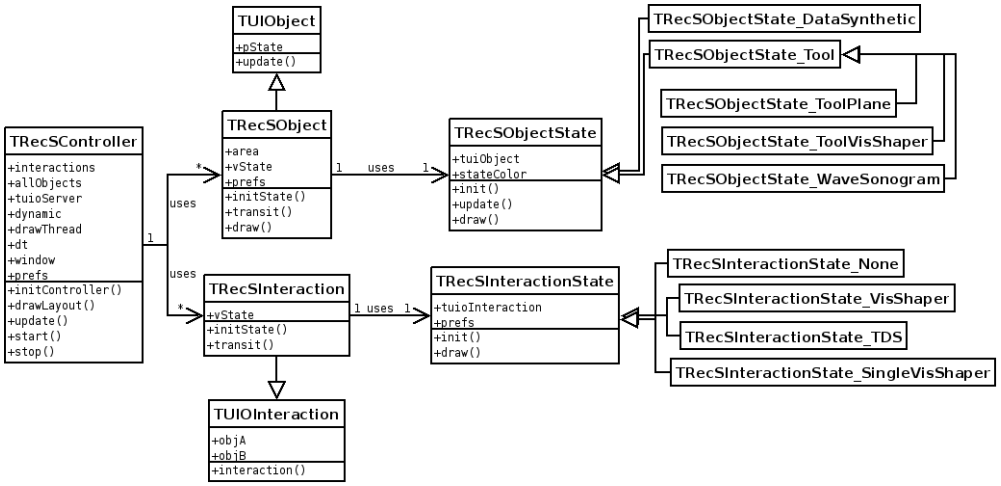


Abbildung 13: Klassendiagramm des Systems

verfahren. Nur wird hier anhand der Typen der beteiligten Objektzustände entschieden, ob und welchen neuen virtuellen Zustand die Interaktion erhält.

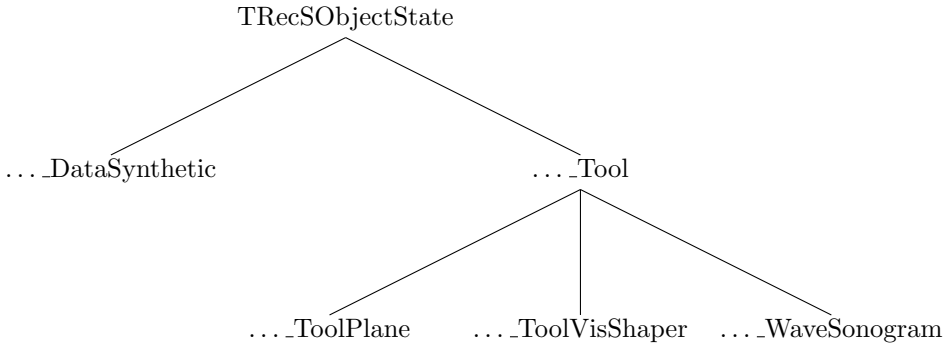


Abbildung 14: Klassenhierarchie der TRecSObjectStates

Klassenhierarchie der TRecSInteractionStates:

Die Funktion `transit()` implementiert mit dieser Hierarchie eine sog. hierarchische State Machine (kurz HSM) [Sam02]. Eine zentrale Instanz der Klasse `TRecSController` (s.u.) iteriert in jedem Zeitschritt durch jede virtuelle Repräsentation der Objekte auf der Tisch und ruft die Funktion `transit()` auf, die das jew. `vState`-Attribut aktualisiert.

Zusätzlich verfügen Objekte der Klassen `TRecSObject`, `TRecSObjectState` und `TRecSObjectInteraction` über die Funktion `draw()`. Diese ist für die graphische Darstellung auf der tangiblen Oberfläche des tDesk zuständig.

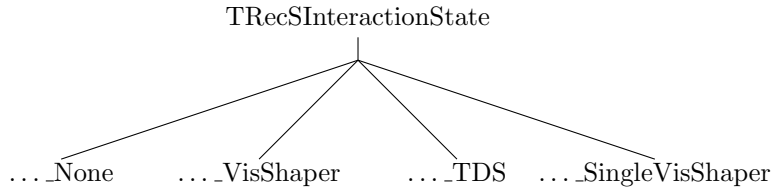


Abbildung 15: Klassenhierarchie der TRecSObjectStates

Verwaltet werden alle Klassen von einer Instanz der Klasse `TRecSController`. Dieser enthält Verwaltungs-Attribute und -Funktionen (z.B. `tuioServer`, `dynamic` und `drawThread`).

Das Attribut `tuioServer` hält eine Instanz des `TUIOServer` aus SETO. Dieser verwaltet die Klassen vom Typ `TRecSObject` und `TRecSInteraction`. Instanzen der Klassen `TRecSObjectState` und `TRecSInteractionState` hingegen werden vom `TRecSController` verwaltet. Dabei wurde die Verarbeitung des zugrunde liegenden Modells (die eigentlichen virtuellen Zustände der Objekte) von der Aktualisierung der graphischen Anzeige in die beiden unabhängigen Threads `dynamic` und `drawThread` aufgeteilt. Daraus resultiert eine geschlossene Interaktionsschleife. Je enger, d.h. je näher sie in Echtzeit arbeitet, desto reaktiver empfindet der Benutzer das System. Es ist uns gelungen, die Latenz zwischen Nutzer-Interaktion und System-Antwort weitestgehend zu minimieren, sodass ein flüssiges Arbeiten möglich wird.

4.2 Datensätze

Um Datensätze in das System zu laden, genügt es ein Objekt in eine der dafür vorgesehenen Ladebuchten zu legen. Ist das Objekt erkannt, wird ein `TRecSObjectState_DataSynthetic`-Objekt erzeugt. Sobald das Objekt in die Interaktionsfläche gestellt wird, ruft die `draw`-Methode des Objekts dessen `load`-Methode auf. Diese bekommt als Parameter unter anderem den Pfad des zu ladenden Datensatzes `dataPath` übergeben. Zudem erhält sie die daraus auszuwählenden Spalten `i` und `j`. Die Variable `dataPath` wird in der `update`-Methode entsprechend der Ladebucht, in der das Objekt liegt, auf einen dort festgelegten Pfad gesetzt. Die Spalten `i` und j werden in der Klasse selbst auf 0 bzw. 1 gesetzt. In der `load`-Methode werden die Daten eingelesen, die Spalten ausgewählt, und auf positive Werte zwischen 0 und 1 normiert. Dies geschieht in der Methode `centerData`. Schließlich wird eine Instanz des `ScatterView` erzeugt, welche einen Scatterplot der Spalten in die Interaktionsfläche zeichnet. In Abb. 16 ist die Darstellung auf der Tischoberfläche dargestellt.

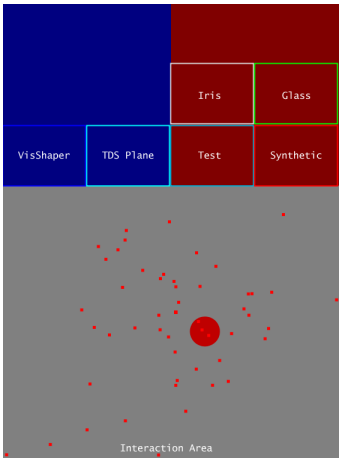


Abbildung 16: Darstellung von Datensätzen auf der TUI

Nachfolgend werden Auszüge der `draw`- und der `load`-Methode aufgeführt.

```
draw {
  if(
  (tuiObject.area == \interaction) && {
    (count == 0)
  },{
    if(dataView == nil ,{
      this.load(rect,dataPath,i,j);
    },{
      count =1;
    });
  }
  );
  ...
}

load{|rect, datapath, i, j|
  dataOriginal = loadPath(dataPath);
  dataOriginal3d = dataOriginal.collect{
    |row| [row[i], row[j], row[2]]};
  data2d = dataOriginal.collect{
    |row| [row[i], row[j]]};
  workData = this.centerData(data2d);
  workData_unmodified_moved = workData.collect{
```

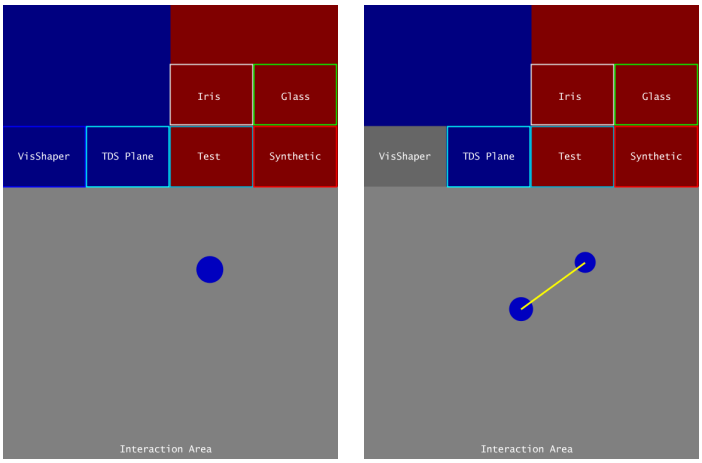
```
      |row| (row - centerPoint)};
    dataView = ScatterView(
      parentWindow, rect, workData, [0, 1].asSpec).resize_(5);
    ...
    dataView.refresh;
    parentWindow.refresh;
}
```

4.3 Tools

Wie im Abschnitt Konzeption beschrieben, kann der Benutzer den Objekten eine Tool-Funktion zugeweisen. Dafür muss er ein Objekt in die entsprechende Ladebox für Tools setzen. Als Ladeboxen stehen ihm dabei die VisShaper-Box und TDS-Plane-Box zur Verfügung. Die Anordnung der Boxen ist in Abbildung 10 zu sehen.

Wenn das Objekt in einer der beiden Ladeboxen steht, bekommt das Attribut `vState` des entsprechenden `TRecSObjects` die Klasse `TRecSObjectState_ToolVisShaper` oder `TRecSObjectState_ToolPlane` zugewiesen und wird somit als Tool deklariert. Beide Klassen sind von `TRecSObjectState_Tool` abgeleitet. Die Klassenhierarchie wurde in Abschnitt 4.1 beschrieben und abgebildet. Die Implementierung der beiden Tools wird in den nächsten beiden Abschnitten näher beschrieben.

4.3.1 VisShaper



(a) Erscheinung des single VisShaper

(b) Erscheinung des double VisShaper

Abbildung 17: VisShaper-Darstellungen auf der TUI

Das VisShaper-Tool dient dazu, die visuelle Repräsentation eines Datensatzes auf der Tischoberfläche zu verändern. Das heißt, dass in Abhängigkeit von der Bewegung des

VisShapers auf dem Tisch die Daten rotiert, verschoben oder gezoomt werden sollen. Dabei unterscheiden wir zwischen dem *SingleVisShaper* und dem *DoubleVisShaper*.

Der SingleVisShaper kommt zum Einsatz, wenn sich mindestens ein Datensatzobjekt und genau ein VisShaperObjekt auf dem Tisch befinden. Ist dies der Fall, so existiert eine TRecSInteraction mit dem vState `TRecSInteractinState_SingleVisShaper`. Die Funktionalität des SingleVisShapers ist in der Methode `interact` der Klasse `TRecSInteractionState_SingleVisShaper` implementiert. In dieser Methode wird die aktuelle Position und Ausrichtung des Fiducial-Markers, sprich des VisShaper-Tools, auf dem Tisch ermittelt. Die Position und die Ausrichtung werden vom TUIO-Server anhand der Lage der Fiducial-Marker bestimmt und als Variablen im TUIObject gespeichert. Wird das VisShaperObjekt auf dem Tisch verschoben oder gedreht, so ändern sich die Werte. Je nach Veränderung dieser Werte sollen die Datenpunkte der Datenobjekte rotiert und/oder verschoben werden. Für die Verschiebung der Daten wird die Methode `interactWithSingleVisShaperTranslate()` aufgerufen. Sie ist eine Methode des Objekts `TRecSObjectState_DataSynthetic` und erwartet als Übergabewert den Verschiebungsvektor des VisShaper-Objekts. Die Methode zieht von allen Datenpunkten den Verschiebungsvektor ab, dadurch werden die Punkte verschoben.

```
workData = workData.collect{|row| row - ([x,y])};
```

Bei der Rotation wird die Methode `rot2dData` aufgerufen, welche ebenfalls eine Methode des `TRecSObjectState_DataSynthetic`-Objekts ist. Als Übergabeparameter bekommt die Methode die Winkeländerung und die Position des `TRecSInteraction`→ `State_SingleVisShaper`. Für die Rotation um einen beliebigen Punkt, in dem Fall um die Position des VisShaper-Objekts, werden die Datenpunkte zunächst um den Positionsvektor verschoben, danach um den Winkel gedreht, und anschließend wieder um den Positionsvektor zurück verschoben.

```
this.moveBy(-1, x, y);
workData = workData.collect{|row| var xData=row[0]; var yData=row[1];
  [
    (cos(diffAngle)*xData)-(sin(diffAngle)*yData),
    (sin(diffAngle)*xData)+(cos(diffAngle)*yData)
  ]};
this.moveBy(1, x, y);
```

Eine Erweiterung des SingleVisShapers stellt der DoubleVisShaper dar. Mit ihm können ebenfalls die Punkte der Datensätze rotiert und verschoben werden und zusätzlich bietet er die Möglichkeit, aus der Visualisierung heraus- oder hinein zu zoomen. Dafür muss ein weiteres VisShaper-Objekt, durch Assoziieren eines Fiducial-Markers in die entsprechende VisShaper-Ladebucht, erzeugt werden. Somit wird eine `TRecSInteraction` erstellt, die als `vState` die Klasse `TRecSInteractionState_`→ `VisShaper` besitzt. Diese Klasse besitzt die beiden Methoden `draw` und `interact`, die bei jeden Durchlauf der Interaktionsschleife aufgerufen werden. Die `draw`-Methode dient dazu, eine virtuelle Verbindungslinie zwischen den beiden VisShaper-Objekten zu zeichnen. In der `interact`-Methode werden analog zum SingleVisShaper zunächst die Positionen der beiden Objekte ermittelt. Aus diesen beiden Positionen berechnet das System die Ausrichtung der Verbindungsgerade und dessen Länge, die für die Rotation

bzw. das Zoomverhalten relevant sind.

Anders als bei SingleVisShaper erfolgt die Rotation nicht mehr unbedingt um ein VisShaper-Objekt, sondern bei gleichzeitiger Bewegung beider VisShaper-Objekte um den Schwerpunkt ihrer Bewegungsänderung. Dieser Rotationspunkt wird in der Methode `computeRotationPoint` berechnet. Somit ergeben sich drei mögliche Fälle, die anschließend nacheinander abgearbeitet werden.

- Rotation
  Wenn sich die Orientierung der Verbindungslinie geändert hat, erfolgt die Berechnungs des Rotationspunktes und ein Aufruf von `rot2Data` des Datensatz-Objekts mit Winkeländerung und Rotationspunkt als Übergabeparameter.

- Verschiebung
  Ändert sich die Position der beiden VisShaper-Objekte gleichzeitig und gleichstark, wird `interactWithDoubleVisShaperTranslate` des Datensatz-Objektes mit der Positionsänderung als Parameter aufgerufen.

- Zoom
  Wenn sich die Länge der Verbingungslinie geändert hat, erfolgt ein Aufruf von `rot2Data` mit Längenänderung als Parameter.

4.3.2 DataScanning

Das Tangible Data Scanning [BHR06a] (TDS) Tool wurde entworfen, um Datensätze akustisch zu explorieren, die in das System geladen wurden. Die Exploration geschieht mit Hilfe einer virtuellen Ebene, welche an der physikalischen Repräsentation verankert und „durch“ die Daten bewegbar ist. Die Ebene erscheint auf dem Tisch als eine Linie, die durch das Objekt verläuft, welches das TDS-Tool repräsentiert. Die Projektion der Ebene auf eine Linie ist ausreichend, da die virtuelle Ebene senkrecht auf der Tischoberfläche steht.

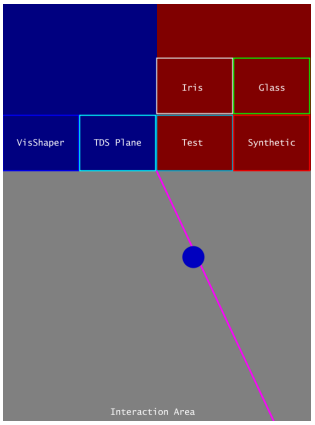


Abbildung 18: Erscheinung des Data-Scanning Tools

Durch die Translation des Tool-Würfels wird natürlich auch die Linie entsprechend bewegt. Außerdem kann der User die Orientierung der Linie im Arbeitsbereich beliebig durch die Rotation des zugehörigen physischen Objektes verändern.

Wird ein Datenpunkt von der virtuellen Ebene durchstoßen, wird er durch sie zum Schwingen angeregt. (→ Model-based Sonification: Spring-Model). Aus solchen hörbarbaren Punkten resultiert eine akustische Datenrepräsentation, die der Benutzer interaktiv in Echtzeit beeinflussen kann.

Die volle Funktionalität der zu Grunde liegenden Sonifikationsmethode, sowie die Berechnung der geschnittenen Datenpunkte, liefert die `TRecSScanning`-Klasse. Sie wurde aus einer schon bestehenden Implementation übernohmen [BHR06a] und für unsere

Zwecke angepasst. Ein `TRecSInteractionState`-Objekt arbeitet auf einer eigenen internen Datenrepräsentation, so wird bei dessen Erzeugung eine Kopie der Originaldaten an die neue Instanz übergeben. Dieser Ansatzt vermeidet unnötigen Datentransfer, und ermöglicht auch besonders große Daten performant zu verarbeiten. Den Einstieg in die Berechnungen bietet die `stepWithRotTrans`-Methode der `TRecSScanning`-Klasse. Sie benötigt die x-, y-Koordinate und den Drehwinkel des TUIO-Objektes.

`TRecSScanning` arbeitet auf einer eigenen internen Datenrepräsentation, die die VisShapertransformationen nicht berücksichtigt. Da die benutzten Berechnungsroutinen die übergebenen Argumente als absolute Koordinaten betrachten, muss eine Koordinatenanpassung erfolgen. Aus dem `vState` des Datenobjektes werden die bereits erfolgten VisShaper-Transformationen durch die `getTransformation`-Methode ermittelt, und in die absoluten Koordinaten umgerechnet.

Die Verklanglichung erfolgt durch einen `klik`-Synth. Es handelt sich um eine additive Mischung eines Pulse- und Sinusoszillators, dessen Anschlagphase durch die Amplitudenhüllkurve hervorgehoben wurde. Die x-Koordinate des angeregten Punktes ist linear auf Panorama gemappt.

4.4 Erweiterbarkeit des Systems

Bis jetzt umfasst unser System nur wenige Tools und Datensätze. Die Anzahl reicht aber für einen Proof of Concept bereits aus.

Das System ist so programmiert, dass es leicht erweitert werden kann. Um ein Tool oder einen Datensatz zu erweitern, muss lediglich ein entsprechender neuer Zustand von `TRecSObjectState`, respektive `TRecSInteractionState` abgeleitet werden. Ist der neue Zustand so definiert, muss lediglich die Zustandsübergangsfunktion `transit()` entsprechend erweitert werden, damit Objekte den State annehemen können. Die Funktionalität der neuen Zustände kann nun mit anderen Objekten auf der Interaktionsfläche genutzt werden.

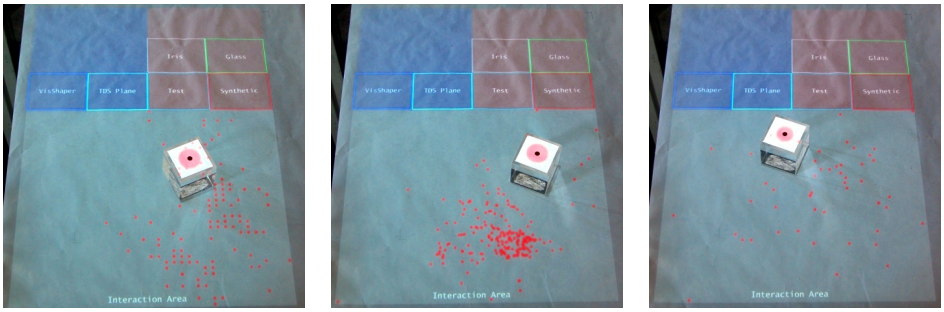
In `TRecSPreferences` sind Listen der vorhandenen Tools `listOfInteractionStates` und deren Interaktionskombinationen `listOfPairsNeededForInteraction` definiert. Erstere enthält die Klassennamen wie sie in `TRecSInteractionState` definiert sind, letztere Listen von in `TRecSObjectState` definierten Klassennamen. Wird dem System ein neues Tool hinzugefügt, so müssen von `TRecSInteractionState` und / oder `TRecSObjectState` entsprechende Klassen abgeleitet, und die oben genannten Listen entsprechend erweitert werden.

5 Anwendungsbeispiele

In vielen Situationen müssen Daten ausgewertet werden. Beispielsweise könnten dies Daten sein, die den bereits besprochenen Datensätzen ähnlich sind, Sportergebnisse der letzten 10 Jahre, Messwerte und Untersuchungsergebnisse in Laboratorien, Umfrageergebnisse oder Auswertungen. Oft sind die anfallenden Daten so umfangreich, dass eine Betrachtung der reinen Zahlen zu keinen Ergebnissen (d.h. Informationsgewinn)

führt. TRecS unterstützt den Anwender bei der Analyse. Die einzelnen Use Cases werden nun erläutert.

Visualisieren eines Datensatzes



(a) Iris (b) Glass (c) Synthetisch

Abbildung 19: Datensätze

Das einfachste Anwendungsbeispiel für das System ist die Anzeige eines Datensatzes als Plot.

Der Benutzer steht vor dem tDesk. Er „lädt“ einen Würfel in der roten Daten-Ladebucht mit einem beliebigen Datensatz. Anschließend platziert er den Würfel in der Interaktionsfläche auf dem Tisch, so dass der Plot des jew. Datensatzes erscheint.

Für den Plot werden bisher lediglich die ersten beiden Merkmale des Datensatzes verwendet. Man kann aber auch eine evtl. vorberechnete Projektion (s. Abschnitt 7) mit Hilfe einer Principle Component Aanalysis [Pea01] oder einer Sammon-Map [SJ69] erstellen und für die Visualisierung nutzen.

Anpassen einer Visualisierung

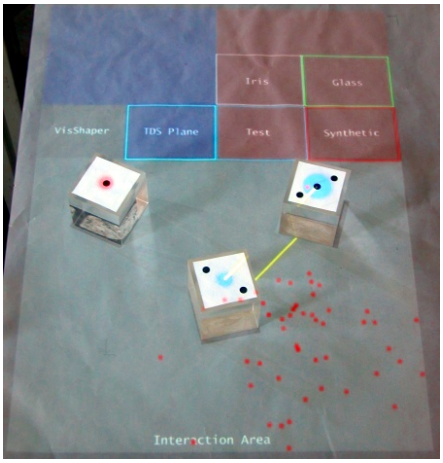
Es kann sein, dass die Darstellung des Plots nicht den Vorstellungen des Benutzers entspricht. Um sie seinen Wünschen anzupassen, wurde das Tool *VisShaper* implementiert. Dieses ermöglicht dem Benutzer, die Visualisierung zu verschieben, zu rotieren oder zu zoomen (vergrößern / verkleinern).

Um den Plot zu rotieren, nimmt der Benutzer einen noch nicht assoziierten Würfel und stellt ihn in der blauen Tool-Ladebucht auf die Fläche des VisShapers, um den Würfel mit diesem Tool zu laden.

Innerhalb der Interaktiosfläche kann der Benutzer durch Drehen dieses VisShaper Objektes den Plot drehen (s. Abb. 20(a)).

Um den Plot zu zoomen wird ein weiteres VisShaper Objekt benötigt, das zusammen mit dem ersten in der Interaktionsfläche arbeitet. Zwischen beiden Objekten wird eine gelbe Linie gezeichnet, die den Status der Visualisierung des Plots repräsentiert (s. Abb. 20(b)). Durch Auseinanderbewegen oder Zusammenschieben der Objekte kann

(a) synthetischer Datensatz und Single-VisShaper



(b) synthetischer Datensatz und Double-VisShaper

Abbildung 20: VisShaper

der Plot vergrößert, oder verkleinert werden. Nach wie vor kann der Plot auch rotiert werden, allerdings werden die Datenpunkte um den Schwerpunkt der Rotation beider Objekte gedreht, um den Plot zu beeinflussen.

Explorieren eines Datensatzes mit Sonifikation

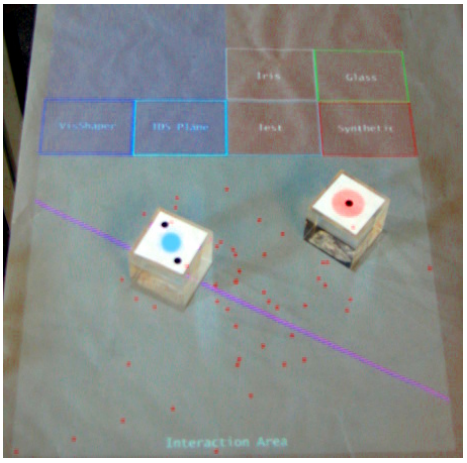


Abbildung 21: Exploration eines Datensatzes mit Sonifikation

Eine vollkommen andere „Sicht“ auf die Datenstruktur bietet das Tool *Tangible Data Scanning* (TDS). Es ermöglicht einem Benuzter in die Daten „reinzuhören“, und diesen Prozess interaktiv zu steuern.

Dafür nimmt der Benutzer einen beliebigen Würfel, welchen er mit der Funktionalität der TDS assoziieren möchte, und stellt es in die entsprechende Ladebucht aus dem blauen Tool-Bereich. In der Interaktionsfläche wird das TDS-Objekt als ein blauer pulsierender Kreis dargestellt. Die magentafarbene Linie, die durch den Kreis verläuft, repräsentiert eine Projektion der Schnittebene der Sonifikation. Jede Berührung der Linie mit einem Datenpunkt wird in einen hörbaren Klang umgewandelt. So kann der Nutzer durch das geschickte Verschieben und Rotieren des TDS-Objektes die für ihn inter-

essanten Datenstellen explorieren.

Akustische Exploration eines Datensatzes bei gleichtzeitiger Anpassung der Visualisierung

Dieses Anwendungsfall war von uns als System-Designer nicht beabsichtigt, als Nutzer aber erwartet. Ein System, das dem Benutzer erlaubt, seine Funktionalitäten so frei wie möglich zu kombinieren, muss Anwendungsfälle, wie dieses (und die folgenden) zulassen.

Die Tools VisShaper und TDS können daher miteinander kombiniert werden. Dabei entsteht eine *emergente* Funktionalität (entsprechend Abschnitt 2.3 „Bedeutung und Kopplung“). Diese tritt auf, wenn die Datenpunkte mit Hilfe des VisShapers verschoben, rotiert oder gezoomt werden, und dabei die Linie des TDS berühren.

Der Nutzer kann somit selbst entscheiden, welche Kombination von Funktionalitäten und wie er mit ihnen umgeht am sinnvollsten und praktikabelsten ist.

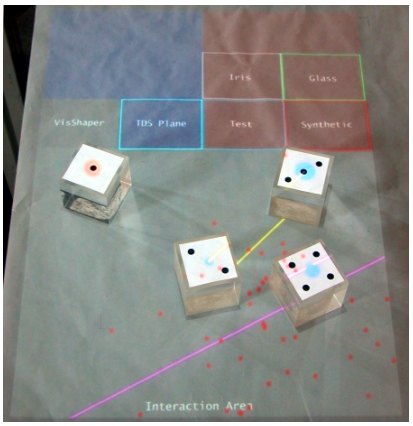


Abbildung 22: Explorieren eines Datensatzes bei gleichtzeitiger Anpassung der Visualisierung

Mehrere DataScanning Objekte

Der Benutzer ist nicht beschränkt in der Anzahl der gleichzeitig verwendeten TDS-Planes, die er zur Datenexploration benötigt. Er kann beliebig viele neue Instanzen des TDS erzeugen, in dem er neuen Objekten die Funktionalität der Sonifikation in der entsprechender Ladebucht zuweist. Jedes Abstellen dieser Objekte in der Interaktionfläche aktiviert ein neues TDS-Tool. Jedes Tool stellt seine eigene virtuelle Schnittebene dar und ist so von anderen TDS-Tools unabhängig. So könnte Der Benutzer auch komplexere „Schnittwände“ konstruieren. Das System ist so ausgelegt, dass der Benutzer mehrere TDS-Objekte gleichzeitig bedienen kann. Berühren die TDS-Linien die Punkte der Datensätze, so werden diese sonifiziert.

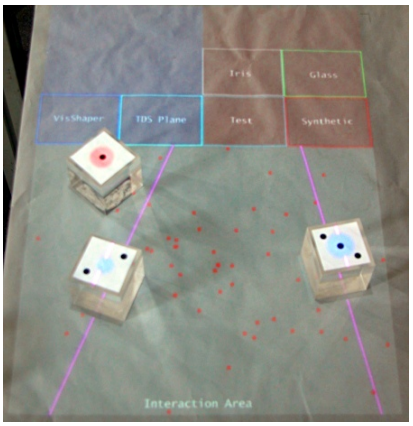


Abbildung 23: Mehrere DataScanning Objekte (mit Datensatz)

Alle Änderungen der Daten, die z.B. durch ein VisShaper erfolgen, wirken in gleichen Maßen auf alle bestehende TDS-Interaktionen.

Mehrere Datensätze

Es besteht die Möglichkeit mit mehreren Datensätzen gleichzeitig zu arbeiten. Dafür muss der User lediglich weitere Würfel mit Datensätzen aufladen und sie in die Interaktionsfläche stellen. Es werden dann alle Datensätze auf dem Tisch geplottet. Nutzt der Anwender nun ein Tool, wie z.B. den VisShaper, so wirken sich die Änderungen auf alle Datensätze gleichzeitig und gleichermaßen aus. Dem Anwender steht dabei frei, welche Datensätze und wie viele er auswählen möchte. Entfernt man einen Würfel, der mit einem Datensatz geladen ist, wieder von der Interaktionsfläche, so wird dieser nicht mehr geplottet. Da die Tools nur mit sichtbaren Datensätzen interagieren können, kann der Anwender jederzeit zwischen den Datensätzen wechseln und somit mit ihnen abwechselnd zusammen oder jeweils allein arbeiten.

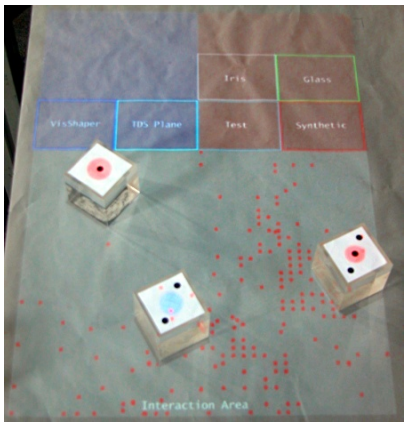


Abbildung 24: Mehrere Datensätze (mit SingleVisShaper)

6 Probleme

Ein Projekt, wie das hier beschriebene, ist auf Grund seiner Komplexität nicht völlig problemlos zu realisieren. Bei der Umsetzung sind diverse Probleme aufgetreten, die hier erwähnt werden sollen.

Da wir uns dafür entschieden haben, den Projektor oberhalb und die Kamera unterhalb der Tischplatte anzubringen, kam es zu einigen Problemen bei der Implementierung der Tools. Wir haben die Projektionsfläche um 90° gedreht und zusätzlich vertikal gespiegelt, um Kamera und Projektor aufeinander abzustimmen. Dies mussten wir bei der Programmierung der Tools immer berücksichtigen, was öfters zu Komplikationen geführt hat.

Problematisch beim Umgang mit dem System ist die Kombination von visuellem Tracken der Fuducial-Marker von unten und der gleichzeitigen Projektion von oben. Die Kamera wird leicht vom Projektor geblendet, sodass eine gleichzeitige Projektion nur bei viel Licht von unten und verhältnismäßig schwacher Projektor-Leistung möglich ist. Der große Nachteil daran ist, dass in dieser Einstellung die Projektion kaum sichtbar ist.

Fast das gesamte Projektteam hatte vorher noch nie Berührung mit der verwendeten Programmiersprache SuperCollider. Weil diese Sprache einige Eigenheiten bei der Syntax hat, z.B. verschachtelte If-Bedingungen, kam es oft zu unerklärlichen Bugs.

Bei der Umsetzung trat die Frage nach einem geeigneten Framework für die Visualisierung auf der Tischoberfläche auf. Dies wurde zunächst versucht, mit der Python-Schnittstelle *PythonSC* von SuperCollider zu bewältigen. Innerhalb von Python wurde auf die *MatPlotLib* gesetzt, welche sich durch einfach zu handhabende und sehr mächtige Zeichen- und Plotfunktionen mit entsprechenden Koordinatentransformationen auszeichnet.

Leider erwies sich MatPlotLib in Kombination mit SuperCollider als sehr inperformant, sodass auf MatPlotLib schließlich verzichtet wurde. Stattdessen wurde begonnen, die nötigen Visualisierungsmethoden mit *Cairo* und *PyGTK* (nach wie vor innerhalb von Python) zu implementieren. Doch auch hier traten Probleme auf. Diese lagen allerdings nicht in der Performanz, sondern in der Tatsache, dass sie sich aufgrund zweier völlig voneinander getrennter Threads (einer in SuperCollider und einer in Python) nicht synchronisieren, bzw. abstimmen ließen. D.h. das Modell des Systems lief in SuperCollider bereits weiter, die Visualisierung in Python konnte jedoch nicht zum Neuzeichnen bewegt werden (trotz entsprechender Kommandos). Dieses Problem ist vermutlich auch bei MatPlotLib aufgetreten, auf Grund der schlechten Performanz aber nicht bemerkt worden.

Schließlich haben wir auf die Standardmethoden von SuperCollider unter Linux zurückgegriffen: *SwingOSC*. Diese realisieren einen in Java geschriebenen, via OSC steuerbaren GUI- und Graphik-Server. Mit diesem ließen sich trotz oder gerade wegen des simplen Kommandoumfangs erstaunlich einfach und robust sämtliche Visualisierungsprobleme lösen.

Die zur Erzeugung der Scatterplots verwendete Klasse `ScatterView` war nicht leicht zu verstehen. So war bei Darstellungsfehlern nicht eindeutig feststellbar, ob der Ursprung des Fehlers im eigenen Code, oder dem des `ScatterView` lag.

Desweiteren erwies sich die Anpassung der „überdeckenden“ Transformation des TDS-Tools an das angezeigte Datenplot als ein hartnäckiges und schwer überschaubares Problem. Das DataScanning Tool verwendet eine eigene interne Datenrepräsentation, die nach jeder Änderung der Visualisation durch entsprechende Rücktransformationen angepasst werden muss. Die Punkte, die nicht im Arbeitsbereich liegen, aber trotzdem am Plotgrenzen angezeigt werden, verhalten sich nicht so in TDS. Die zum Teil noch widersprüchliche Informationen entziehen der Darstellung die Eindeutigkeit.

Ein anderes Problem tauchte beim Umgang mit größeren Datensätzen auf. So kann es passieren, dass das System bei Benutzung des DataScanning Tools deutlich langsamer wird. Dies geht zu Lasten der Reaktivität des Systems. Wird die Latenz zwischen Nutzer-Interaktion und System-Antwort zu groß, droht an dieser Stelle, die Interaktionsschleife aufzubrechen.

7 Zusammenfassung & Abschluss

Es ist uns gelungen, ein neuartiges (single-user) System zu entwickeln, das dem Benutzer eine nicht-spezialisierte tangible Arbeitsumgebung bietet.

Gemäß dem Konzept hat der User die freie Wahl bei der Nutzung der Tools und Datensätze. Diese können ohne Einschränkungen ausgewählt und miteinander kombi-

niert werden. Dem Benutzer wird dabei nicht vorgegeben, in welcher Reihenfolge oder Anzahl die Objekte erstellt werden müssen.

Ein frei konfigurierbares Baukastensystem war Ziel des Projektes, dass somit erreicht und durch die vorhandene und vorher nicht angedachte emergente Funktionalität sogar übertroffen wurde.

Anzumerken wäre noch, dass wir den Explorationsprozess von beliebigen tabellarischen Daten im Sinne des Tangible Computings in einem tangiblen Interface umsetzen konnten. Die Daten sind allerdings weiterhin nicht direkt „anfassbar“. Dafür aber der Explorationsprozess im Sinne der Zielsetzung. Man könnte argumentieren, dass das tangible Interface des Systems durch ein reines Multitouch Interface [Han] ohne physische Repräsentationen ausgetauscht werden könnte. Jedoch unrterstützt gerade diese die Wahrnehmung der Daten durch das auftretende Coupling.[Dou04].

Ideen für weitere Entwicklungen

Das von uns entwickelte Baukastensystem bietet einen universellen Rahmen für Entwicklungen und Erweiterungen auf dem Gebiet Tangible Computing zur explorativen Datenanalyse. Daher ist es auch prädestiniert, um bereits vorhandene Entwicklungen, wie Konzepte aus dem AmbiD [BHR06c] oder die TI-Son [HBRR07] in dieses System wieder einfließen zu lassen, um dem Benutzer ein breiteres Spektrum an Analysetools zu bieten.

Beide Systeme dienen der Analyse von Echtzeit-Daten. Das AmbiD bietet dem Anwender die Möglichkeit, diverse Datenquellen in verschiedene Datensenken zu lenken. Dabei wurde die räumliche Beziehung auf dem Tisch ausgenutzt, bei der die Nähe der Objekte auf die Signalstärke der zwischen den Quellen und Senken fließenden Daten aufmultipliziert wurde, um eine Art impliziten Fader für jede Verbindung zu gewinnen.

Im TI-Son System wurde spezielles Augenmerk auf die tangible Steuerung und Ausnutzung der räumlichen Beziehung der Objekte mit Übertragung dieser Beziehung auf ein räumliches Audio-Setup (8-Kanal Surround) gelegt. D.h. die Fläche des tDesk repräsentiert einen virtuellen Raum, in dem Objekte platziert werden, die jede für sich einen Datenkanal darstellt. Relativ zu einem sog. Listener-Objekt, welches ebenfalls in diesen Raum gestellt wird, erklingen Sonifikationen der einzelnen Kanäle mit Hilfe des räumlichen Audio-Setups aus entsprechenden Richtungen.

Diese Konzepte bieten sich an, um im TRecS System ein Redesign zu erfahren.

Eine zentrale, noch fehlende Funktionalität ist die Möglichkeit, Datensätze zu manipulieren und in neuen Ladebuchten speichern zu können. Sobald dies möglich ist, ergeben sich viele weitere Erweiterungsmöglichkeiten:

interaktive Exploration einer Sammon Map [SJ69]
: Es ist denkbar, dass ein spezielles Tool entwickelt wird, das einen hochdimensionalen Datensatz mit einer Sammon Map-Projektion in eine niedrigdimensionalen Datensatz überführt. Dieser könnte dann mit vorhandenen Tools exploriert werden.

interaktive Exploration einer SOM, HSOM, etc. [Koh82], [OR01]

Dasselbe wie bei der Sammon Map-Exploration könnte auch bei SOM basierten Datensatztransfomationen angewendet werden.

Simulated Annealing [DKN87]
lokales „Schmelzen“ und „Abkühlen“ (z.B. zum semi-automatischen Routen von Leiterbahnen auf Platinen).

Dimensionsauswahltool ein Tool zur Auswahl der zwei Spalten eines Datensatzes, die im Plot dargestellt werden sollen.

Abgesehen von Datensatzmanipulationen sind natürlich noch andere Erweiterungen möglich. So ließen sich analog zum DataScanning weitere interaktive modell-basierte Sonifikationen, z.B. Particle Trajectory oder Data Wave Sonogram [HR], einführen.

Auch die Art der Datensätze lässt sich bedingt erweitern. Datenbanken eignen sich z.B. auch zur Speicherung von Medien, wie Sound-, Video oder Bildmaterial. Mit speziellen Anzeige-Tools wäre es dann möglich, diese ebenfalls im TRecS System zu explorieren. Als Inspiration könnte Abschnitt fünf in [Ull02] dienen.

Ein weiterer wichtiger Aspekt für Verbesserungen ist die Multiuser-Fähigkeit des Systems. Designt wurde es für einen Benutzer. Tangible Interfaces bieten aber häufig die Möglichkeit, dass mehrere Benutzer simultan mit ihnen arbeiten. Die Benutzer würden dabei stark vom Konzept der Kollaboration profitieren. Dafür müsste die einseitige Ausrichtung des TUI aufgehoben und von allen Seiten zugänglich gemacht werden. Eine Möglichkeit dies zu erreichen wäre es, die Ladebuchten von der Interaktionsfläche zu entkoppeln und ähnlich wie bei VoodooSketch [BHG+07], Paletten mit Ladebuchten einzuführen. Somit wäere gewährleistet, dass jeder Benutzer einen gleichberechtigten Zugang zu den Buchten und der Interaktionsfläche hätte. Für das Paletten-Konzept müsste aber das Tracking-System maßgeblich verändert oder die verwendeten Objekte mit VoodooIO-Bausteinen versehen werden.

Darüber hinaus ist es lediglich eine Frage der Phantasie und des Könnens des Interface-Designers, Erweiterungen für das System zu schreiben.

Um den guten Ruf von Tangible Computing in Bezug auf Nützlichkeit und Nutzbarkeit zu untersuchen, wäre es denkbar, das System in soziologischen Experimenten weiter zu untersuchen. Die Ergebnisse dann könnten zur weiteren Verbesserung wieder zurück in das Design fließen.

Danksagung

Wir danken Dr. Thomas Hermann und Till Bovermann für die Initialisierung des Projektes, die ausführliche Einführung in die Materie und ausdauernde Unterstützung bei der Umsetzung.

Quelltext-Segmente

Listing 1: Grundcode der Basisklasse TRecSObject

```
TRecSObject : TUIObject {

  // virtual state information
  // about the current object instance
  var <>vState;

  *new {|format, id|
    ^super.new(format,id).initState;
  }

  initState {
    vState = TRecSObjectState(this);
  }

  // inherited by TUIObject, called by TUIO_OSCServer
  update {
    vState.update;
  }

  // Called by TRecSController each time frame
  transit {
    // Determines if a state change is necessary
    // and performs this.
  }

  // Called by TRecSController
  draw {|window, cam2scrWidth, cam2scrHeight|
    GUI.pen.use{
      // Here drawing takes place
    };
  }
}
```

Listing 2: Grundcode der Basisklasse TRecSObjectState

```
TRecSObjectState {

  var <>tuiObject; // the corresponding TUIO

  *initClass {
  }

  *new {|tuiObject|
    ^super.new.init(tuiObject);
  }
```

```
11
12    init{|argObject|
13      tuiObject = argObject;
14    }
15
16    // Called by TRecSController
17    update {
18    }
19
20    // Called by TRecSController
21    updateInvisible {
22    }
23
24    // Called by TRecSController
25    draw {|window, cam2scrWidth, cam2scrHeight|
26      GUI.pen.use{
27        // Here drawing takes place
28      };
29    }
30
31    removeStuff{
32    }
33  }
```

Listing 3: Grundcode der Basisklasse TRecSInteraction

```
1   TRecSInteraction : TUIOInteraction {
2
3     var <>vState;
4
5     *new {|objA, objB|
6       ^super.new(objA, objB).initState;
7     }
8
9     initState {
10      vState = TRecSInteractionState_None(this);
11    }
12
13    // Called by TRecSController
14    transit {
15    }
16
17    // inherited by TUIObject, called by TUIO_OSCServer
18    interaction {
19      vState.interact;
20    }
21  }
```

Listing 4: Grundcode der Basisklasse TRecSInteractionState

```
TRecSInteractionState {

  var <tuioInteraction;
  var <>allVisObjects;
  var <>knownObjects;

  *new {|tuioInteraction|
    ^super.new.init(tuioInteraction);
  }

  init{|argInteraction|
    tuioInteraction = argInteraction;
    allVisObjects = [];
    knownObjects = [];
  }

  // Called by TRecSController
  draw {|window, cam2scrWidth, cam2scrHeight|
  }

  interact {
  }
}
```

Literatur

[art08] ARToolKit Home Page, 2008. http://www.hitl.washington.edu/artoolkit/.

[BHG+07] F. Block, M. Haller, H. Gellersen, C. Gutwin, and M. Billinghurst. Voodoosketch: Physical Interface Palettes and Sketched Controls alongside Augmented Work Surfaces. 2007.

[BHR06a] T. Bovermann, T. Hermann, and H. Ritter. Tangible data scanning, 2006. http://www.techfak.uni-bielefeld.de/ags/ni/projects/datamining/datason/demo/ICAD2006/TDSSon.html.

[BHR06b] T. Bovermann, T. Hermann, and H. Ritter. Tangible data scanning sonification model. pages 77–82, London, UK, 2006. Department of Computer Science, Queen Mary, University of London, UK.

[BHR06c] T. Bovermann, T. Hermann, and H. Ritter. A tangible environment for ambient data representation. In D. McGookin and S. Brewster, editors, *First International Workshop on Haptic and Audio Interaction Design*, volume 2, pages 26–30, Glasgow, UK, 2006. www.multivis.org.

[BKJ05] R. Bencina, M. Kaltenbrunner, and S. Jorda. Improved Topological Fiducial Tracking in the reacTIVision System. *Proc. of the IEEE Int. Workshop on Projector-Camera Systems (PROCAMS 2005)*, 2005.

[BMHS03] R. Berry, M. Makino, N. Hikawa, and M. Suzuki. The Augmented Composer Project: The Music Table. In *Proceedings of the 2003 International Symposium on Mixed and Augmented Reality*, Tokyo, Japan, 2003.

[Bov07] T. Bovermann. Seto, 2007. `http://tuio.lfsaw.de/seto.shtml`.

[DKN87] F. Darema, S. Kirkpatrick, and V. A. Norton. Parallel algorithms for chip placement by simulated annealing. *IBM Journal of Research and Development*, 31(3):391–402, 1987.

[Dou04] P. Dourish. *Where the Action Is : The Foundations of Embodied Interaction (Bradford Books)*. The MIT Press, September 2004.

[gla87] UCI Machine Learning Repository: Glass Identification Data Set, 1987. `http://archive.ics.uci.edu/ml/datasets/Glass+Identification`.

[Han] J.Y. Han. Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection.

[HBRR07] T. Hermann, T. Bovermann, E. Riedenklau, and H. Ritter. Tangible computing for interactive sonification of multivariate data. 2007.

[Her] T. Hermann. sonification.de - definition of sonification. `http://sonification.de/main-def.shtml`.

[Her08] T. Hermann. Taxonomy and definitions for sonification and auditory display. 2008.

[HHR04] T. Hermann, T. Henning, and H. Ritter. Gesture desk - an integrated multi-modal gestural workplace for sonification. 2915:369–379, 2004.

[HR] T. Hermann and H. Ritter. Listen to your data: Model-based sonification for data analysis. *Advances in intelligent computing and mulimedia systems*, pages 189–194.

[iri88] UCI Machine Learning Repository: Iris Data Set, 1988. `http://archive.ics.uci.edu/ml/datasets/Iris`.

[iri08] Iris flower data set - Wikipedia, the free encyclopedia, 2008. `http://en.wikipedia.org/w/index.php?title=Iris_flower_data_set&oldid=181069391`.

[IU97] H. Ishii and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, 1997.

[JKGB05] S. Jordà, M. Kaltenbrunner, G. Geiger, and R. Bencina. The reacTable*. In *Proceedings of the International Computer Music Conference (ICMC 2005)*, Barcelona, Spain, 2005.

[KBBC05] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza. TUIO: A protocol for table-top tangible user interfaces. *Proc. of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[Miz] M. Mizutani. www.michihito.com : Tasting Music. `http://www.michihito.com/11tastingmusic.html`.

[OR01] J. Ontrup and H. Ritter. Hyperbolic Self-Organizing Maps for Semantic Navigation. *Advances in Neural Information Processing Systems*, 14:1417–1424, 2001.

[Pea01] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

[PRI02] J. Patten, B. Recht, and H. Ishii. Audiopad: A Tag-based Interface for Musical Performance. In *Proceedings of the 1st Conference on New Interfaces for Musical Expression (NIME02)*, Dublin, Ireland, 2002.

[Sam02] M. Samek. *Practical Statecharts in C/C++: Quantum Programming for Embedded Systems*. CMP, 2002.

[sc07] Supercollider, 2007. `http://supercollider.sourceforge.net/`.

[SJ69] JW Sammon Jr. A Nonlinear Mapping for Data Structure Analysis. *Computers, IEEE Transactions on*, 100(18):401–409, 1969.

[swi07] SwingOSC Readme, 2007. `http://www.sciss.de/swingOSC/`.

[UI97] B. Ullmer and H. Ishii. The metaDESK: models and prototypes for tangible user interfaces. *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 223–232, 1997.

[UI99] J. Underkoffler and H. Ishii. Urp: a luminous-tangible workbench for urban planning and design. *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pages 386–393, 1999.

[UIG98] B. Ullmer, H. Ishii, and D. Glas. mediaBlocks: physical containers, transports, and controls for online media. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 379–386, 1998.

[Ull02] B. Ullmer. Tangible interfaces for manipulating aggregates of digital information. 2002.