

# Second Hand Interface

Eine Vergleichstudie zweihändiger Interaktionsmodelle

## **Bachelor Report**

Tobias Lensing  
SoSe 2006 , Digitale Medien B.Sc.  
Fachbereich 3: Informatik  
Universität Bremen

Erstgutachter: Prof. Jürgen Friedrich  
Zweitgutachter: Matthias Krauss

# Inhalt

<b>1</b>	<b>Abstract</b>	<b>4</b>
1.1	English Version	4
1.2	Deutsche Fassung	4
<b>2</b>	<b>Einleitung</b>	<b>5</b>
<b>3</b>	<b>Die Testanwendung</b>	<b>6</b>
3.1	Konzept	6
3.1.1	Szenario	6
3.1.2	Die beiden Interaktionsmodelle	7
3.1.2.1	Intention	7
3.1.2.2	Symmetrisches Modell	7
3.1.2.3	Asymmetrisches Modell	9
3.1.3	Grafische Darstellung der Mauszeiger	11
3.1.4	Begründung der getroffenen Entscheidungen	11
3.1.4.1	Belegung der Maustasten	11
3.1.4.2	Darstellung der Mauszeiger im symmetrischen Modell	12
3.1.5	Weitere Eigenschaften der Testanwendung	12
3.2	Implementierung	13
3.2.1	Einleitung	13
3.2.2	Eine Übersicht über die Architektur des bestehenden Frameworks	13
3.2.3	Einbinden der Hardware	16
3.2.4	Modifikationen und Integration von CPNMouse	17
3.1.5	Modifikation des bestehenden Frameworks	20
3.1.5.1	Identifikation und Verwaltung von mehreren Zeigegeräten	20
3.1.5.2	Modifikationen an User Interface und Mausnachrichten	20
<b>4</b>	<b>Die Studie</b>	<b>22</b>
4.1	Methode	22
4.1.1	Rahmenbedingungen	22
4.1.2	Aufgabe	22
4.1.2.1	Aufgabenstellung	22
4.1.2.2	Anforderungen an die Aufgabenstellung	22
4.1.3	Prozedur	23
4.1.4	Probanden	24
4.1.5	Erwartungen	24

4.2	Evaluation	25
4.2.1	Quantitative Evaluation	25
4.2.1.1	Messung	25
4.2.1.2	Vergleichskriterien	26
4.2.1.3	Auswertung der Messdaten	27
4.2.1.4	Ergebnisse	27
4.2.1.5	Analyse	28
4.2.1.5	Bemerkung zur Analyse	33
4.2.2	Qualitative Evaluation	33
4.2.2.1	Datenerhebung	33
4.2.2.2	Auswertung	33
4.2.2.3	Ergebnisse	34
4.2.2.4	Besonderheiten und Kommentare	35
4.3	Diskussion und Beurteilung	35
4.4	Interpretation im Hinblick auf Guiard	38
<b>5</b>	<b>Fazit</b>	<b>38</b>
<b>6</b>	<b>Literaturverzeichnis</b>	<b>39</b>
<b>APPENDIX I:</b>	<b>Der Fragebogen</b>	<b>40</b>
<b>APPENDIX II:</b>	<b>Screenshot der Testanwendung</b>	<b>41</b>
<b>APPENDIX III:</b>	<b>Hinweise zu der beiliegenden CD</b>	<b>41</b>

# 1 Abstract

## 1.1 English Version

Two different bimanual interaction models were compared in a study. For both models two identically constructed commercial USB mice were used as input devices. In the first model the same functionality was assigned to both input devices. This approach is called the *symmetrical* model. In the second model different functionality was assigned to the input devices so that the nondominant hand of the user got another function than the dominant one. In this so-called *asymmetrical* model an adapted variant of the *Toolglass* technique was used that is well-known from other studies, scientific papers and applications. For the realization of the study a test application was created. For evaluating the study results both data measured by that application and a questionnaire completed by the test users were used. Primary aim of the study was to compare both models in respect of acceptance, intuitiveness, precision and efficiency. Interpretation of the study results showed that both model's performance turned out to be quite similar, but the symmetrical model had some advantages in acceptance, intuitiveness and efficiency.

## 1.2 Deutsche Fassung

Es wurden zwei spezielle zweihändige Interaktionsmodelle in einer Studie verglichen. Bei beiden dienten zwei baugleiche, handelsübliche optische USB-Mäuse als Eingabegeräte. Im ersten Modell wurde beiden Eingabegeräten die gleiche Funktionalität zugeordnet; dieser Ansatz wird als das *symmetrische* Modell bezeichnet. Im zweiten Modell wurden beiden Eingabegeräten unterschiedliche Funktionen zugeordnet, so dass die nichtdominante Hand des Benutzers eine andere Aufgabe übernahm als die dominante. In diesem als *asymmetrischen* Modell bezeichneten Interaktionsmodell wurde eine adaptierte Variante der so genannten *Toolglass*-Technik verwendet. Für die Durchführung der Studie wurde eine Testanwendung erstellt. Zur Auswertung wurden aufgezeichnete Messdaten sowie ein den Testnutzern ausgehändigter Fragebogen herangezogen. Ziel der Studie war es im Wesentlichen, Akzeptanz, Intuitivität, Präzision und Effizienz beider Modelle miteinander zu vergleichen. Die Interpretation der Studienergebnisse zeigte auf, dass beide Modelle ähnliche Ergebnisse hervorbrachten, der symmetrische Ansatz jedoch im Hinblick auf Intuitivität, Effizienz und Akzeptanz vorteilhafter war.

## 2 Einleitung

Seit der Erfindung der Computermaus durch Douglas C. Engelbart und der Einführung grafischer Benutzeroberflächen [Smith, D. C. (1982)], hat sich der typische PC-Arbeitsplatz dahingehend entwickelt, dass der Nutzer zusätzlich zur Tastatur im Normalfall genau ein Zeigegerät – meistens die Maus – verwendet. Aufgrund der großen Verbreitung dieses *Status quo* haben sich die Benutzer solcher Systeme an bestimmte Handhabungen bzw. Bedienabläufe gewöhnt. Als Konsequenz hieraus haben Softwarehersteller ihre Produkte, die einen großen Anwenderstamm erreichen sollen, auf die Eigenschaften jener Standardsysteme abgestimmt. Ich möchte wie andere Interfacedesigner [Beaudouin-Lafon, M. (2000)] die These vertreten, dass die herkömmlichen einhändig bedienbaren WIMP<sup>1</sup>-Interfaces immer mehr an ihre Grenzen stoßen. Die von Myers und Buxton [Buxton, W. (1986)] sowie die von Kabbash, Buxton und Sellen [Kabbash, P. (1994)] durchgeführten Studien unterstützen diese These. Sie zeigen auf, dass zweihändige Interaktionsmodelle gegenüber dem Status quo effizientere Ergebnisse hervorbringen können, ohne dass Anwender intensiv hierfür trainiert werden müssen.

Im Gegensatz zu den oben erwähnten Studien möchte ich nicht versuchen, zweihändige Interaktionsmodelle mit dem herkömmlichen einhändigen Ansatz zu vergleichen. Vielmehr möchte ich zwei besondere zweihändige Modelle gegenüberstellen. Hiermit möchte ich feststellen, inwiefern die beiden Modelle von den Testnutzern akzeptiert werden, wie präzise, effizient und intuitiv sie sind und wie gut die nichtdominante Hand in die Bedienabläufe integriert wird. Für die Durchführung der Studie habe ich eine Testanwendung entwickelt, die ein typisches alltägliches Anwendungsszenario implementiert, welches für eine beidhändige Bedienung prädestiniert ist: die Transformation von Objekten auf einer zweidimensionalen Arbeitsfläche. Dieses Szenario ermöglicht es dem Benutzer in einer visuellen, objektorientierten Umgebung "Gegenstände" *direkt* zu manipulieren ("direct manipulation" [Schneiderman, B. (1983)]). Dieses Prinzip ist von grafischen Benutzeroberflächen hinreichend bekannt; die beidhändigen Interaktionsmodelle ermöglichen es jedoch, die Direktheit der Manipulation weiter zu maximieren. Die entwickelte Testanwendung könnte man somit als mit einem so genannte Post-WIMP User Interface ausgestattet bezeichnen.

Im Folgenden werde ich zuerst detailliert auf das Konzept und die Besonderheiten der Implementierung der Testanwendung eingehen. Darauf folgend beschreibe ich die Durchführung der Studie selbst sowie die Auswertung der Studienergebnisse.

---

1 Windows, Icons, Menus & Pointers (WIMP)

# 3 Die Testanwendung

## 3.1 Konzept

### 3.1.1 Szenario

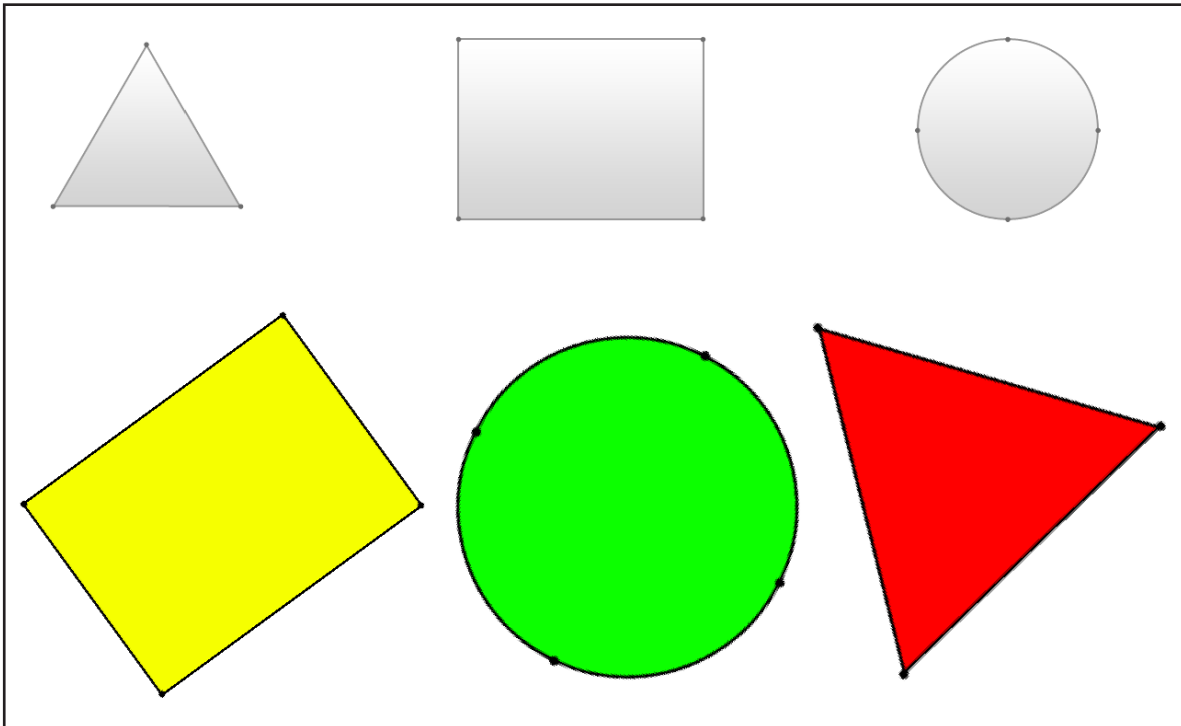


Abbildung 1: Arbeitsfläche der Testanwendung mit Objekten

Die Testanwendung, die speziell für die Studie entwickelt wurde, sollte ein einfaches Szenario realisieren, bei dem der Testbenutzer verschiedene Objekte auf einer zweidimensionalen Arbeitsfläche platzieren musste. Um bei der quantitativen Auswertung der Studie nicht in eine zu komplexe Struktur zu geraten, sollten die Benutzer eine möglichst einfach mess- und überprüfbare Aufgabe absolvieren, die für jeden verständlich und zugleich gut vergleichbar bleiben sollte. Ich entschied mich dazu dem Benutzer eine der folgenden Anforderung ähnliche Aufgabe<sup>2</sup> zu stellen:

»Auf dem Bildschirm sind drei Referenzobjekte zu sehen, die gerade positioniert auf der Arbeitsfläche liegen, grau gefärbt sind und jeweils eine andere Form repräsentieren. Weiterhin

<sup>2</sup> für die letztendliche Aufgabenstellung der Studie siehe 4.1.2.1

befinden sich drei rotierte bzw. skalierte Objekte auf der Arbeitsfläche, die durch verschiedene Formen und Farben gekennzeichnet sind. Transformieren Sie die farbigen Formen so, dass sie möglichst präzise auf den Referenzobjekten liegen.«

Die Aufgabe sollte von den Studienteilnehmern sowohl mit dem symmetrischen als auch mit dem asymmetrischen Bedienmodell durchgeführt werden. Abbildung 1 zeigt eine exemplarische Form der Arbeitsfläche, die den Testnutzern präsentiert wurde.

### 3.1.2 Die beiden Interaktionsmodelle

#### 3.1.2.1 Intention

Wie bereits erwähnt sollen in dieser Studie zwei besondere zweihändige Bedienmodelle miteinander verglichen werden. Das erste Modell wird als *symmetrisch* bezeichnet, da beide Hände die gleiche Funktion übernehmen, das zweite als *asymmetrisch*, da jede Hand eine andere Funktion übernimmt.

Die Arbeit von Guiard [Guiard, Y. (1987)] legt nahe, dass der Großteil aller zweihändiger Aufgaben es mit sich bringt, dass beide Hände völlig verschiedene Aufgaben übernehmen, und dass die menschliche Motorik und Kognition auf dieses Szenario besonders gut eingestellt ist. Ich möchte nun – zusätzlich zu der bereits in der Einleitung erklärten Absicht – überprüfen, ob dies für das gegebene Testszenario ebenfalls gegeben ist, indem ich ein bekanntes asymmetrisches zweihändiges Interaktionskonzept, die Toolglass-Technik [Bier, E. A. (1993)], mit einem auf einer Metapher aufbauenden symmetrischen Konzept vergleiche.

#### 3.1.2.2 Symmetrisches Modell

Im symmetrischen Modell sind beide Mauszeiger »gleichberechtigt«. Bei beiden Mäusen ist stets ausschließlich die jeweils linke Maustaste mit einer Transformationsfunktionalität belegt. Die rechte sowie die mittlere Maustaste bleiben in diesem Modell funktionslos. Daher werde ich im Folgenden nur noch von »der Maustaste« einer Maus sprechen, die als die linke Maustaste der jeweiligen Maus zu verstehen ist.

Beiden Eingabegeräten bleibt in diesem Modell beim Klicken und Bewegen stets die gleiche Funktionalität zugeordnet. Der Benutzer hat lediglich die Möglichkeit zwischen zwei verschiedenen Bearbeitungsmodi zu wählen: nur translieren<sup>3</sup> oder frei transformieren.

Der erste Modus ist implizit immer dann aktiv, wenn der Benutzer ein Objekt nur mit einer Maus manipu-

---

<sup>3</sup> verschieben

liert. Klickt der Benutzer mit einer einzigen Maus auf ein Objekt und hält die Maustaste gedrückt, so kann er das Objekt durch Verschieben der Maus auf den zwei Dimensionen der Arbeitsfläche translieren. Durch Loslassen der Maustaste wird das Objekt ebenfalls wieder »losgelassen«, also auf der Arbeitsfläche platziert. Die zweite Maus kann während dieser Aktion ungenutzt bleiben oder gleichzeitig für die Translation eines anderen Objekts verwendet werden.

Der zweite Modus ist immer dann aktiv, wenn für die Manipulation eines Objekts beide Mäuse gleichzeitig benutzt werden. Klickt der Benutzer mit beiden Mäusen auf ein Objekt und hält die Maustasten gedrückt, so verwandeln sich die Mauszeiger zu kleinen Punkten. Durch Bewegen beider oder einer Maus können nun alle möglichen verschiedenen Transformationen auf das Objekt ausgeübt werden. Die Transformation des Objekts wird so berechnet, dass die vom Benutzer ausgewählten Referenzpunkte im relativen Raum des Objekts immer an der Stelle bleiben, an welcher der Nutzer die Maustaste heruntergedrückt hat. So hat er – durch Bewegung beider oder einer der Mäuse – die Möglichkeit ein Objekt zu rotieren, uniform zu skalieren und zu translieren. Lässt er eine der Maustasten los, so verbleibt das Objekt in dem letzten Transformationszustand auf der Arbeitsfläche.

Bei beiden Modi passt die Metapher des »Anfassens« und »Loslassens«. Vergleichen könnte man die Art der Manipulation, einmal abgesehen von der uniformen Skalierung, mit einem Blattpapier, das vor einem auf einem Tisch liegt. Das Blattpapier repräsentiert in diesem Fall eine der Formen und der Tisch die Arbeitsfläche in der Testanwendung. Mit beiden Zeigefingern berührt man nun zwei Punkte auf dem Papier. Bewegt man den rechten Zeigefinger in einer kreisförmigen Bahn um den linken, so wird das Papier um den Rotationspunkt, den man mit dem linken Zeigefinger gewählt hat, gedreht. Umgekehrt könnte man auch den linken um den rechten Zeigefinger drehen. Durch eine gleichförmige zusätzliche Bewegung beider Hände in eine bestimmte Richtung lässt sich eine gleichzeitige Translation des Papiers auf dem Tisch ausführen. Ebenso ließe sich das Papier nur bewegen, entweder nur mit einer oder mit beiden Händen. In dem symmetrischen Modell der Testanwendung hat der Benutzer zusätzlich die Möglichkeit eine uniforme Skalierung auszuüben, in dem er, um weiter in dieser Metapher zu sprechen, beide Finger auf dem Papier voneinander weg zieht. Die folgende Abbildung visualisiert die beschriebenen Vorgänge:



Abbildung 2: Gleichzeitige Translation zweier Objekte mit zwei Mäusen

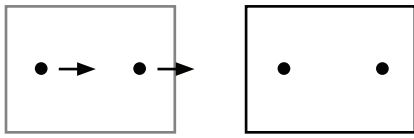


Abbildung 3: Translation eines Objekts mit zwei Mäusen

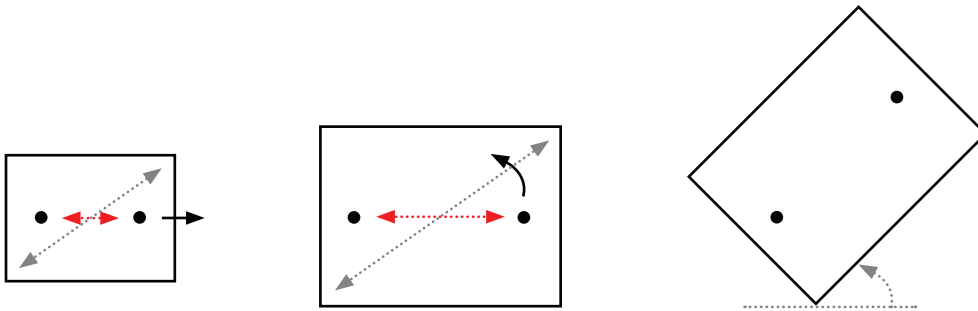


Abbildung 4: Skalierung (Schritt 1 zu 2) und Rotation (Schritt 2 zu 3) mittels zwei Mäusen auf einem Objekt

### 3.1.2.3 Asymmetrisches Modell

Im asymmetrischen Modell erhält jede Maus eine spezielle Funktion. In diesem Modus wird zwischen dominanter und nichtdominanter Hand unterschieden. Im Normalfall des Rechtshänders übernimmt also die rechte, dominante Hand eine andere Aufgabe als die linke, nichtdominante. In dem asymmetrischen Interaktionsmodell griff ich auf ein im Zusammenhang mit beidhändigen Benutzeroberflächen populär gewordenes Prinzip zurück, das Toolglass. Bier, Stone, Pier, Buxton und DeRose beschreiben in Ihrem Paper "Toolglass and Magic Lenses: The See-Through Interface" [Bier, E. A. (1993)] dieses Konzept der zweihändigen Interaktion detailliert. In der Testanwendung habe ich letztendlich eine adaptierte Version des ursprünglichen Toolglass-Konzepts verwendet, welches optisch und funktional jedoch den Eigenschaften des von Bier, et. al. beschriebenen Prinzips ähnelt, jedoch nur einen Teil des beschriebenen Funktionsspektrums, die *Click-through Buttons* nutzt.

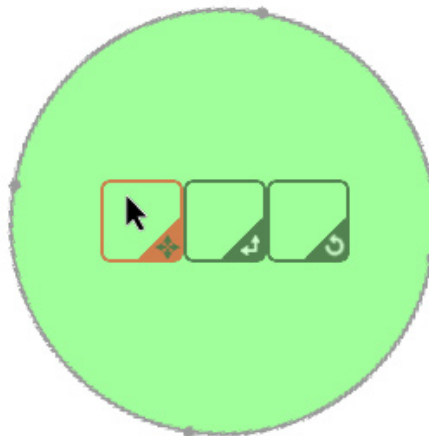


Abbildung 5: Toolglass der Testanwendung

Abbildung 5 zeigt die grafische Darstellung des Toolglass in der Testanwendung, optisch angelehnt an die von Bier, et. al. beschriebene Konzeption. Das an die Anforderungen der Anwendung angepasste Toolglass stellt lediglich drei verschiedene Click-through Buttons zur Verfügung. Der linke dient der Translation, der mittlere der uniformen Skalierung und der rechte der Rotation des Objekts auf der Arbeitsfläche. Somit erhält der Benutzer die selben Kontrollmöglichkeiten wie auch schon im symmetrischen Modell.

Das Toolglass selbst wird – wie von Bier, et. al. beschrieben – durch Bewegung der Maus, die durch die nichtdominante Hand gesteuert wird, positioniert. Die Maustasten dieser Maus sind hier funktionslos. Um die Funktionen der Click-through Buttons zu nutzen, muss der Benutzer das Toolglass nun über einem der eingefärbten Objekte platzieren. Mit der Maus, die von seiner dominanten Hand gesteuert wird, klickt er die linke Maustaste über der gewünschten Schaltfläche an und hält sie herunter gedrückt. Durch Bewegung dieser Maus kann er das unterliegende Objekt nun translieren, skalieren oder rotieren, je nach dem welchen Click-through Button er ausgewählt hat. Lässt er die Maustaste wieder los, so verharrt das Objekt in der veränderten Lage auf der Arbeitsfläche. Während der Benutzer die Objekte mit der dominanten Hand manipuliert, kann er das Toolglass als Hilfsmittel zeitgleich mit der linken Maus weiter bewegen. Die folgende Tabelle zeigt einige Besonderheiten der im Toolglass befindlichen Click-through Buttons in der Testanwendung auf:

Click-through Button	Reaktion auf Mausbewegung der dominanten Hand
Translation	Verschieben der Maus auf der X-Achse entspricht verschieben der X-Position des Objekts auf der Arbeitsfläche (horizontale Translation); Verschieben der Maus auf der Y-Achse entspricht verschieben der Y-Position des Objekts auf der Arbeitsfläche (vertikale Translation)
Skalierung	Verschieben der Maus auf der Y-Achse nach oben entspricht Vergrößerung, nach unten Verkleinerung. Als Referenzpunkt dient der angeklickte Punkt auf dem Objekt.
Rotation	Verschieben der Maus auf der Y-Achse nach oben entspricht der Erhöhung, nach unten der Verringerung des Rotationswinkels. Als Referenzpunkt dient hier ebenfalls der angeklickte Punkt auf dem Objekt.

### 3.1.3 Grafische Darstellung der Mauszeiger

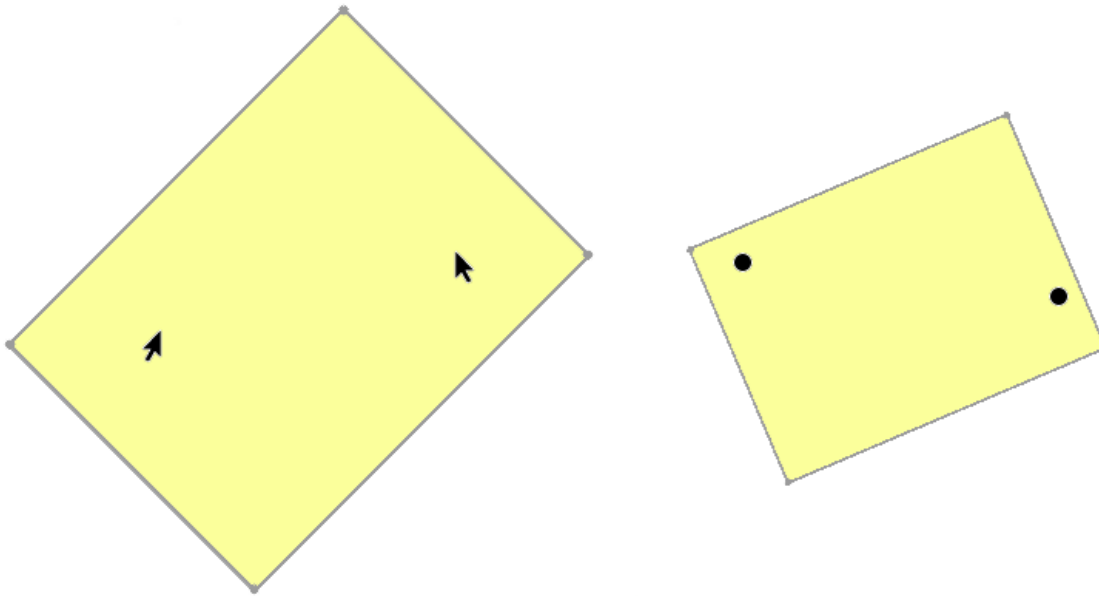


Abbildung 6: Verschiedene Mauszeiger (Darstellung rechts nur im symmetrischer Modus)

Abbildung 6 zeigt die verschiedenen Mauszeiger. Im linken Bild ist die Ausgangsdarstellung der Zeiger zu sehen. Die linke Maus besitzt einen pfeilartigen Zeiger, der nach rechts zeigt. Umgekehrt zeigt jener der rechten Maus nach links. Das rechte Bild zeigt die Darstellung der beiden Mauszeiger wenn der Benutzer ein Objekt im symmetrischen Modus transformiert. Während des Transformationsprozesses werden beide Zeiger durch einen Punkt dargestellt. Die Darstellung im asymmetrischen Modus kommt vollständig mit der im linken Bild zu sehenden Anzeige aus (vgl. Abb. 5).

### 3.1.4 Begründung der getroffenen Entscheidungen

#### 3.1.4.1 Belegung der Maustasten

Insbesondere die Belegung der Maustasten festzulegen, stellte sich als diffizil heraus. Für die rechte Maus ist jeder Anwender daran gewöhnt, dass die linke Maustaste eine Primärfunktion auslöst. Für die linke Maus gibt es jedoch keinen solchen Konsens. Hier stellte sich die Frage, ob für die linke Maus nun auch die linke oder doch eher die rechte Maustaste diese Primärfunktionalität übernehmen sollte. Schließlich ist die natürlichere Haltung der linken Hand so ausgelegt, dass der Zeigefinger eher auf der rechten Maustaste liegt als auf der linken. Im Rahmen der durchgeführten Studie stellte sich heraus, dass diese Entscheidung weder korrekt noch inkorrekt war: es liegt schlicht und ergreifend am jeweiligen Benutzer, welche Maustaste er für die Primärfunktion bei der linken Maus als intuitiv ansieht.

### 3.1.4.2 Darstellung der Mauszeiger im symmetrischen Modell

Die Darstellung der beiden Mauszeiger im symmetrischen Modell wurde teils aus Gründen der Wahrnehmbarkeit, teils aus Intuition festgelegt. In der Ausgangssituation (der Benutzer hat nicht mit beiden Mäusen ein Objekt »angefasst«) habe ich mich bewusst gegen eine farbliche Unterscheidung der beiden Mauszeiger entschieden, da diese sonst womöglich mit der Farbgebung der Objekte selbst interferiert hätten. Bei der Gestaltung der Mauszeiger während der freien Transformation eines Objekts (Punktendarstellung) könnte bemängelt werden, dass der Nutzer keinerlei visuellen Anhaltspunkt mehr hat, welcher Zeiger zu welchem physischen Eingabegerät gehört. Es zeigte sich jedoch, dass die Zuordnung der Zeiger zu den Mäusen allein durch die Beobachtung der Bewegungen in den meisten Fällen nachvollziehbar ist.

### 3.1.5 Weitere Eigenschaften der Testanwendung

Zusätzlich zur Realisierung des Testszenarios selbst muss die Testanwendung über ein System zur Messung<sup>4</sup> der Benutzereingaben verfügen. Dieses Testsystem erzeugt Binärdateien, die die Benutzereingaben detailliert protokollieren. Aus diesen Binärdateien lassen sich später durch ein externes Werkzeug CSV-Dateien<sup>5</sup> erzeugen, die mit Tabellenkalkulationsprogrammen wie Microsoft Excel oder OpenOffice Calc auswertbar sind.

Des Weiteren müssen einige Kontrollfunktionen zur Verfügung gestellt werden, die dazu dienen die Umgebungsparameter des Tests festzulegen:

- Festlegen des Links- oder Rechtshändermodus
- Festlegen des aktuellen Interaktionsmodells (symmetrisches oder asymmetrisches Modell)
- Aktivieren des Capturing-Modus für die Messung der Benutzereingaben

Diese Kontrollfunktionen wurden in Form von kleinen Schaltflächen in einer Art Statusleiste am unteren Bildschirmrand platziert und nicht von den Probanden selbst bedient. Da Statusleisten aus dem täglichen Umgang mit modernen grafischen Benutzeroberflächen hinreichend bekannt sind und aufgrund ihrer Position und Größe kaum wahrgenommen werden, stören diese Kontrollelemente den eigentlichen Testfall nicht.

---

4 Die genaue Messung der Benutzereingaben wird in 4.2.1.1 erläutert.

5 Comma Separated Value (CSV) Files [Wotsit.org (2006)]

## 3.2 Implementierung

### 3.2.1 Einleitung

Aufgrund meiner Programmiererfahrung mit Microsoft Windows XP, habe ich die Testanwendung für dieses Betriebssystem entwickelt. Als Programmiersprache wählte ich C/C++<sup>6</sup>. Für die Implementierung nutzte ich ein bereits aus einem früheren Projekt entstandenes Anwendungs-Framework, welches eine recht flexible und schnelle Basis für ein GUI<sup>7</sup> unter Verwendung von Microsoft<sup>®</sup> Direct3D<sup>®</sup> sowie der Win32 API<sup>8</sup> zur Verfügung stellt. Die Entscheidung dieses Framework zu nutzen traf ich, um für die mit zwei Zeigegeräten verbundenen Anpassungen im System selbst so flexibel wie möglich sein zu können. Für das Ansprechen zweier getrennt agierender Zeigegeräte unter Windows war es notwendig einen speziell hierfür konzeptionierten Treiber zu benutzen.

Bei der Implementierung der Anwendung zeigte es sich, dass sehr tiefgreifende Modifikationen im Framework selbst notwendig waren, um das gewünschte Anwendungsszenario zu realisieren. Während sich die Anpassungen im Bereich der Infrastruktur zur Nachrichtenübertragung innerhalb der GUI-Systems selbst in Grenzen hielten, stellte sich das Ansprechen und Einbinden der Hardware in das bestehende Framework als äußerst kompliziert dar. Im Folgenden erläutere ich zuerst den relevanten Part der Architektur des bestehenden Frameworks, bevor es an die zweihändige Bedienung angepasst wurde. Danach zeige ich die Details der Einbindung mehrerer voneinander getrennter Zeigegeräte in das Framework auf.

### 3.2.2 Eine Übersicht über die Architektur des bestehenden Frameworks

Das bestehende Framework, das ich verwendet habe, um die Anwendung zu realisieren, stellt eine Basisbibliothek – genannt *TLib* – für allgemeine Low-Level-Operationen (Input/Output, Verwaltung von Dynamic Link Libraries (COM<sup>9</sup>), Threads und (XML-)Datenstrukturen) sowie eine erweiterte Bibliothek – genannt *Trinity* – für High-Level-Operationen von User interfaces (Windows- und Messaging-System und die zugehörigen Präsentationsalgorithmen) zur Verfügung. Zur Anzeige verwendet diese Bibliothek wie bereits oben erwähnt Direct3D. Dies gewährleistet eine schnelle, hardwarebeschleunigte Anzeige sowohl von 2D- als

---

6 C/C++ eignete sich insbesondere deshalb, weil das bestehende Framework sowie die für das Ansprechen der Zeigegeräte Schnittstelle ebenfalls auf C++ bzw. C basierten.

7 Graphical User interface (GUI)

8 Microsoft Win32 Application Programming Interface

9 Component Object Model

auch von 3D-Objekten. Grob handelt es sich bei dem gesamten Framework um ein an die MFC<sup>10</sup> angelehnte Klassenbibliothek, die jedoch durch ein integriertes COM und ein hochleistungs 3D-Rendering-System erweiterte Funktionen sowie eine größere Flexibilität gewährleistet.

Konzeptioniert ist die High-Level-Bibliothek für die Anzeige und Verwaltung von dreidimensionalen User Interfaces. Zweidimensionale User Interfaces werden in den 3D-Raum projiziert und Pixelgenau dargestellt. So entsteht der Eindruck eines "echten" zweidimensionalen Interfaces. Die Vorteile der hardwaremäßigen 3D-Beschleunigung und des Managements durch die Direct3D-Bibliothek bleiben jedoch erhalten. Hierdurch eignet sich *Trinity* insbesondere für die Realisierung der Testanwendung, da die Umsetzung des Test-szenarios – die Transformation von Objekten auf einer zweidimensionalen Arbeitsfläche – größtenteils auf Aufgaben basiert, die bereits durch *Trinity* oder eine der unterliegenden Bibliotheken (*TLib* bzw. Direct3D) gelöst sind. *Trinity* nutzte vor der Anpassung an die Bedürfnisse der Testanwendung für diese Studie mehrere Threads, um die Anzeige durch das System und Manipulation durch den Benutzer voneinander zu trennen. Abbildung 7 stellt die *Trinity*-Architektur vereinfacht dar:

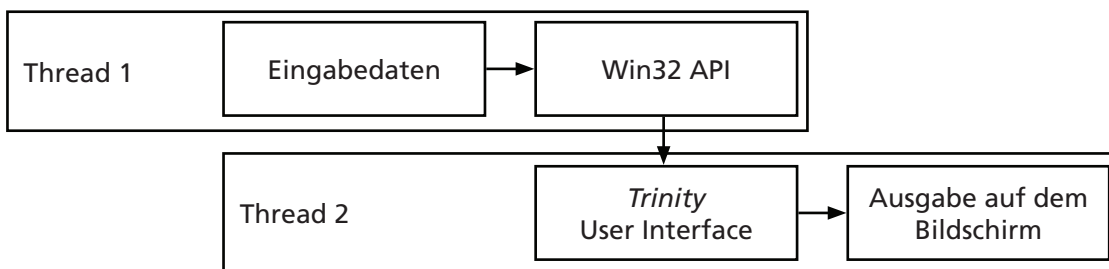


Abbildung 7: Threads in einer vereinfachten Architektur der Trinity-Bibliothek

Zentrales Element des *Trinity* User Interface, welches für die Umsetzung der Anwendung und die notwendigen Änderungen an der Bibliothek selbst von Belang ist, ist im Wesentlichen die Klasse *CTWindow*, die instanziiert ein Fensterobjekt innerhalb des User Interface repräsentiert. Jedes Fensterobjekt repräsentiert dabei ein rechteckiges, zweidimensionales Objekt.

class CTWindow	Attribute	Nachrichten
	Matrix scale	MOUSEDOWN_L
	Matrix rotate	MOUSEUP_L
	Matrix translate	MOUSEOVER
	Matrix present	MOUSELEAVE
	Window parent	MOUSEMOVE
WindowSet children		

Abbildung 8: Die relevanten Attribute und Nachrichten der CTWindow-Klasse.

10 Microsoft Foundation Class Library [Microsoft Corp., MFC (2006)]

Abbildung 8 zeigt die für die Anwendung relevanten Attribute und Nachrichten der Klasse *CTWindow*. Die drei Matrixattribute *scale*, *rotate* und *translate* dienen hauptsächlich zur Zwischenspeicherung der einzelnen Transformationsschritte (Skalierung, Rotation sowie Translation) für ein Fensterobjekt; die Matrix *present* ist das Ergebnis der Multiplikation dieser drei Matrizen:

$$present = scale * rotate * translate$$

*present* beinhaltet also die aktuelle Transformationsmatrix eines Fensterobjekts im Raum des jeweiligen Elternfensters. Jedes Fenster in dem System beinhaltet somit seinen eigenen Unterraum. Durch zwei weitere Attribute (*parent* und *children*) entsteht eine Baumstruktur von verschachtelten Fensterobjekten.

Die Nachrichten entsprechen denen eines herkömmlichen Fenstersystems, ähnlich wie bereits in dem Messaging-System der Win32 API implementiert. Die folgende Tabelle erläutert die Bedeutung der in Abbildung 8 aufgelisteten relevanten Nachrichten:

Nachricht	Beschreibung
MOUSEDOWN_L	Herunterdrücken der linken Maustaste
MOUSEUP_L	Loslassen der linken Maustaste
MOUSEOVER	Mauszeiger wurde in den Bereich des Fensters bewegt
MOUSELEAVE	Mauszeiger hat den Bereich des Fensters verlassen
MOUSEMOVE	Mauszeiger wurde innerhalb des Fensterbereichs bewegt

Wenn ein durch den Benutzer über die Maus eingeleitetes Ereignis eintritt, wird von einer zentralisierten Klasse *CTUserInterface* das gewünschte Fenster ermittelt und die entsprechende Nachricht an das jeweilige Fensterobjekt geschickt. Verknüpft mit der Nachricht (z.B. MOUSEDOWN\_L) selbst übermittelt das Framework – je nach Nachrichtentyp – bestimmte weitere Informationen. In der vor der Anpassung an die Anwendung vorliegenden Version war dies für alle Mausnachrichten die relative Position der Maus im Unterraum des Fensters. In dieser Version konnte das Framework lediglich ein Zeigegerät, da zur Erfassung der Eingabeereignisse die Win32 API verwendet wurde. Abbildung 3 zeigt eine vereinfachte Version des ursprünglichen Ablaufs der Ereignis- und Nachrichtenverwaltung:

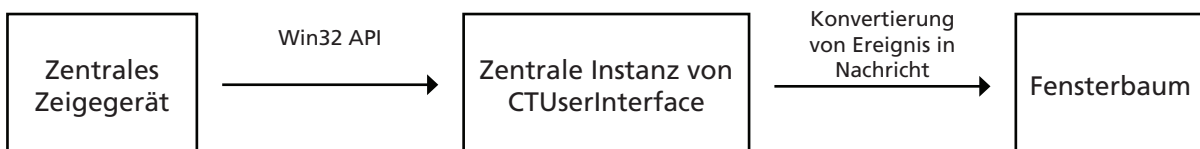


Abbildung 9: Die Nachrichtenverwaltung in der ursprünglichen Version des Frameworks

Aus Abbildung 9 wird ersichtlich, dass zur Ansteuerung von zwei voneinander getrennten Zeigegeräten einige strukturelle Veränderungen in der Architektur des Frameworks notwendig werden. Bevor ich die überarbeitete Architektur erläutere, gehe ich im Folgenden auf die Einbindung der Hardware auf der unter dem Framework liegenden Ebene ein, um die Art und Weise der Anpassungen zu begründen.

### 3.2.3 Einbinden der Hardware

Wie bereits erwähnt war es notwendig, einen speziellen Treiber für das Ansprechen mehrerer Zeigegeräte unter Windows XP zu verwenden. Das Betriebssystem bietet leider von sich aus keine vollständige Unterstützung hierfür an. Es ist zwar durchaus möglich, mehrere USB-Mäuse an ein Windows XP® System anzuschließen, die herkömmlichen USB-Maustreiber verwalten mehrere physische Geräte in diesem Fall aber nicht als solche in der Software. Das heißt, dass mehrere angeschlossene Hardwaregeräte zu einem virtuellen Softwaregerät zusammengefasst werden; ein Zeiger wird also von mehreren Mäusen gesteuert. Für Software, die oberhalb des Kernelmode abläuft sind die beiden physischen Geräte somit auf "normalem" Wege nicht mehr voneinander zu unterscheiden [Westergaard, M. (2002)]. Allerdings bieten auch Technologien wie die Microsoft DirectInput API<sup>11</sup>, die vorwiegend von Spielesoftware eingesetzt werden, keine Abhilfe. Microsoft selbst spricht in der Dokumentation zum DirectX 9.0c SDK zwar einen Ausweg [Microsoft Corp., DirectInput (2005)] an; dieser führt jedoch über einen nachträglich in die Win32 API implementierten und ausschließlich unter Windows XP zur Verfügung stehenden Umweg (WM\_INPUT Message) und stellte sich zudem als kompliziert und schlecht dokumentiert heraus. Diesen Ansatz möchte ich hier deshalb bewusst unbeachtet lassen.

Als relativ komfortabel stellte sich die von Michael Westergaard implementierte Schnittstelle, CPNMouse [CPNMouse (2006)] dar. CPNMouse basiert auf zwei Komponenten: zum einen ein Treiber, der anstelle des Originaltreibers für jede beliebige USB-Dreitastenmaus installiert werden kann; zum zweiten eine Programmierschnittstelle (API), die sich in beliebige auf C/C++ basierende Anwendungssoftware integrieren lässt und zum Ansprechen des Treibers – sozusagen an der Win32 API vorbei – dient. Zusätzlich beinhaltet CPNMouse eine Beispielanwendung, in der dessen Funktionalität demonstriert wird. Westergaard stellt seine CPNMouse-Lösung kostenlos als OpenSource auf der Internetplattform sourceforge.net [SourceForge.net (2006)] zur Verfügung und veröffentlicht zusätzlich eine wissenschaftliche Ausarbeitung zu dem Softwarepaket [Westergaard, M. (2002)], in der detailliert aufgezeigt wird, wie das System genau funktioniert. Aufgrund der einfach erscheinenden Architektur und der Tatsache, dass es sich um eine völlig offen implementierte Schnittstelle handelt, entschied ich mich dazu, Westergaards Lösung zu verwenden, um die zwei Zeigegeräte anzusprechen.

---

11 Application Programming Interface (API)

Westergaards API besteht aus zwei voneinander getrennten Modulen: die High- und die Low-Level-API. Die Low-Level-API ist lediglich dazu da, Eingabedaten des Treibers aus dem Kernel heraus in den Speicher der eigentlichen Anwendung zu befördern. Seine High-Level-API bietet eine Funktionsbibliothek für die Umrechnung der Eingabedaten in Mauspositionen, die Übersetzung der konvertierten Eingabedaten zurück in Windows Messages sowie die Darstellung der Mauszeiger im Windows-Betriebssystem. Ein Kernprinzip von CPNMouse ist die unbedingte Nutzung von Multithreading, was bei der Einbindung in die Testanwendung später zu Problemen führte. Dies werde ich im folgenden Abschnitt detaillierter erläutern.

### 3.2.4 Modifikationen und Integration von CPNMouse

Da die Darstellung der Mauszeiger in der für die Studie entwickelten Anwendung nicht über die von Westergaard genutzte GDI-Schnittstelle<sup>12</sup>, sondern über das Fenstersystem der *Trinity*-Bibliothek selbst realisiert werden sollte, entfernte ich die entsprechende Funktionalität aus der CPNMouse High-Level-API. Danach integrierte ich die CPNMouse API in die Trinity-Bibliothek. Der Ablauf des Datentransfers von der Zeigegeräthardware zur Testanwendung wird in Abbildung 10 dargestellt.

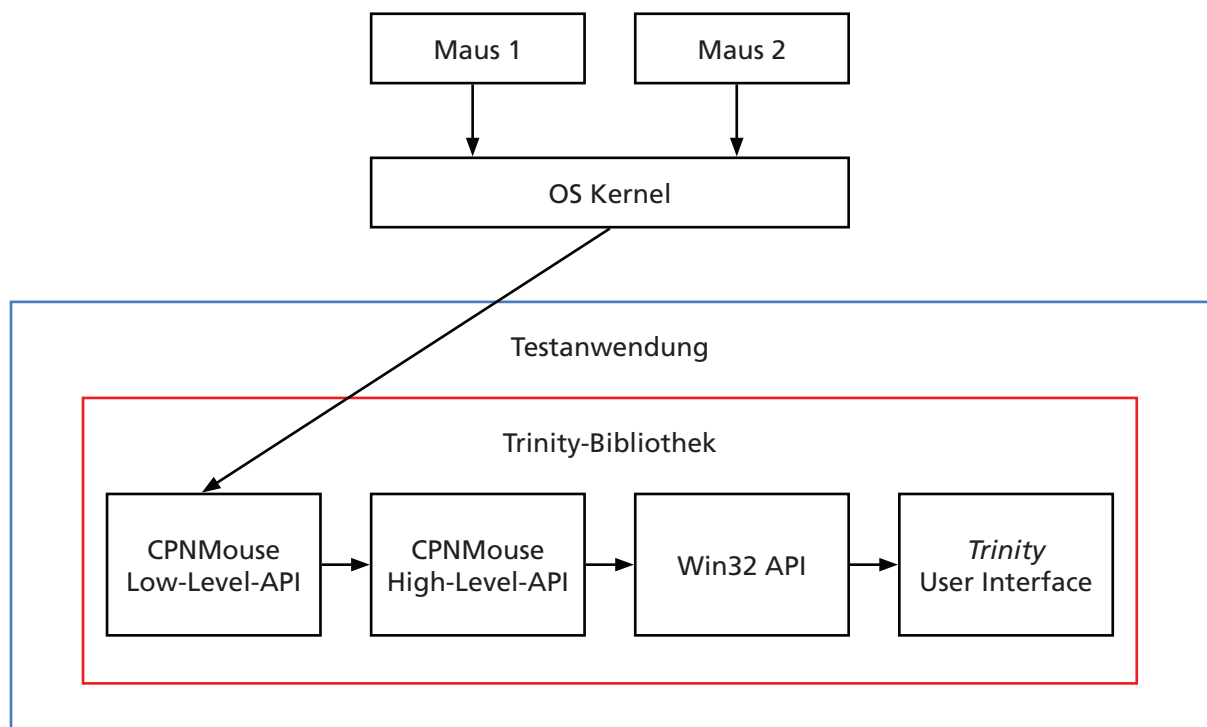


Abbildung 10: Datenübermittlung von Zeigegeräthardware zu Testanwendung

12 Microsoft Graphics Device Interface

Wie aus Abbildung 10 ersichtlich wird, verwaltet auch CPNMouse die Eingabeereignisse der eingebundenen Zeigergeräte wieder über die Win32 API. Dazu nutzt es einen eigenen Management-Thread, der die Eingabedaten aus dem Kernel sammelt (unter Verwendung der Low-Level-API), sie in eine nutzbare Form umrechnet, in Windows Messages konvertiert und schließlich über die SendMessage()-Funktion der Win32 API an ein vorher festgelegtes Fenster der Anwendung schickt. Von dort aus werden die Eingabedaten dann an ein Verwaltungsobjekt innerhalb der *Trinity*-Bibliothek geschickt, welche sie dann nutzt, um das User Interface der Testanwendung zu aktualisieren.

Probleme gab es hier weniger im Hinblick auf den strukturellen Ablauf der Datenübermittlung als vielmehr im Zusammenhang mit der Lastenverteilung zwischen den verschiedenen Threads. Wie bereits auf Seite 6 erwähnt (und in Abbildung 1 dargestellt), nutzte die ursprüngliche Version der *Trinity*-Bibliothek einen eigenen Thread für die Darstellung des User Interface. Dieser aktualisierte im Hintergrund ständig dessen grafische Darstellung. Durch die Integration von CPNMouse in diese Architektur entstand nun ein Problem: der Thread von CPNMouse, der für die Aktualisierung der Mauseingabedaten verantwortlich ist, bekam nicht genügend Prozessorzeit, um für eine »flüssige« Mauseingabe zu sorgen, da der Rendering-Thread<sup>13</sup> diese verbrauchte. Das Resultat war ein unnatürliches »Nachziehen« der Mauszeiger, welches besonders unerfreuliche Ausprägungen zeigte, wenn beide Mäuse schnell und gleichzeitig bewegt wurden. Dieses Problem war im Voraus nicht ersichtlich, da der Beispielanwendung im Lieferumfang von CPNMouse die gesamte Prozessorzeit zum Aktualisieren der Mauszeiger zur Verfügung stand.

Lösen wollte ich dieses Problem, indem ich den Rendering-Thread, der permanent die Anzeige der grafischen Benutzeroberfläche aktualisierte, aus dem Programmablauf entfernte. Stattdessen sollte der CPNMouse-Thread selbst bei Eintritt eines Eingabeereignisses die Anzeige des User Interface aktualisieren. Nach Durchführung dieser Modifikation war das Problem widererwartend immer noch nicht gelöst. Immer noch kam der Prozessor nicht mit der »flüssigen« Aktualisierung der Mauszeiger-Positionen hinterher. Es stellte sich heraus, dass schlicht und ergreifend zu viele Eingabeereignisse durch das Framework gesendet wurden, so dass der Rendering-Thread nicht schnell genug die Kontrolle über den Prozessor an die CPNMouse-Bibliothek zurück gab.

Die endgültige Lösung des Problems bestand darin, einen Highperformance-Timer in den Programmablauf zu integrieren<sup>14</sup>. Dieser Timer beinhaltet einen Framelimiter, der dafür sorgt, dass nur eine bestimmte Anzahl von Frames pro Sekunde<sup>15</sup> durch die Rendering-Routinen erzeugt und auf dem Display des Benutzers

---

13 zur Darstellung des grafischen User Interface

14 Ähnlich wie in Spieleprogrammcode genutzt, um Animationsabläufe von der Prozessorgeschwindigkeit zu entkoppeln.

15 In der finalen Version der Testanwendung wurde die Anzahl von max. dargestellten Frames pro Sekunde auf 30 festgelegt.

dargestellt werden. Dadurch bekommen die CPNMouse-Routinen genügend Zeit, um alle Eingabedaten in Echtzeit zu verarbeiten, wodurch wieder eine flüssige Mauszeigerbewegung entsteht.

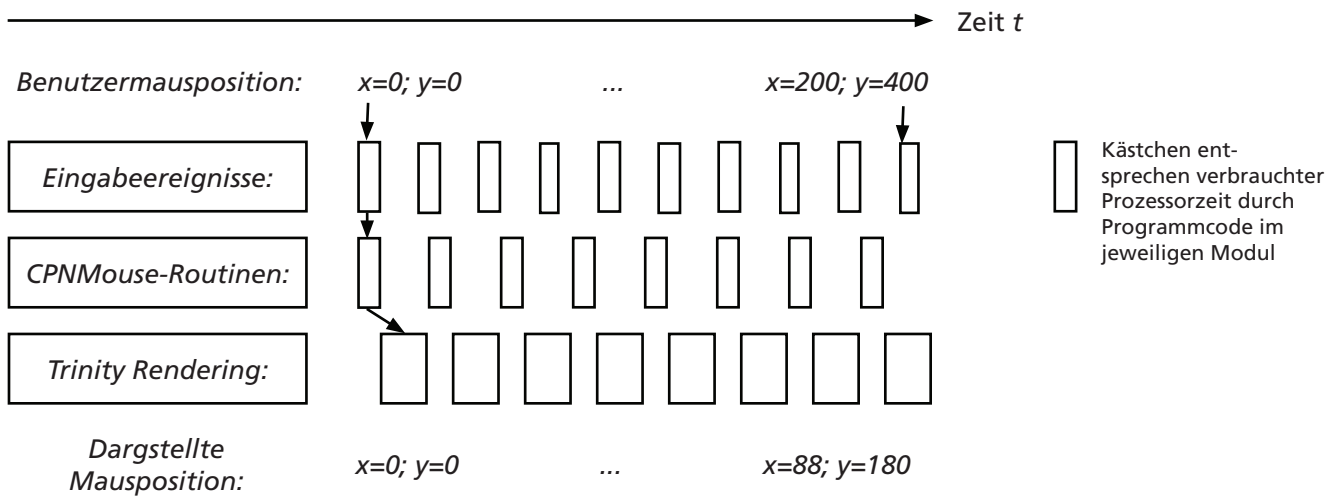


Abbildung 11: Verzögerung durch den Trinity-Rendering Programmcode

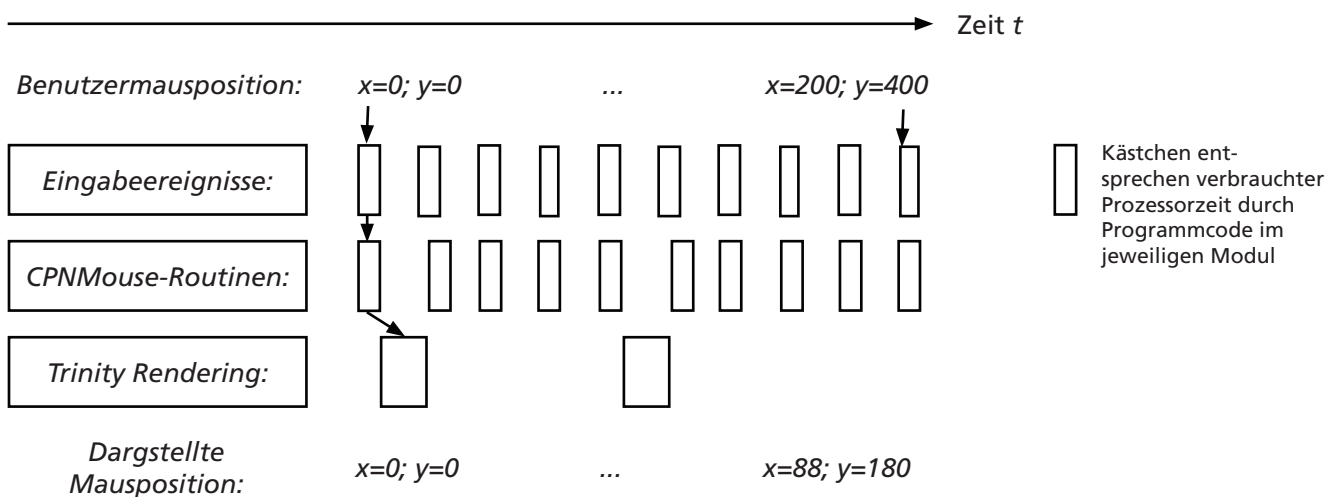


Abbildung 12: Kompensation durch Timer / Framelimiter

Abbildung 12 zeigt die Aufteilung der Prozessorzeit in der mit dem Framelimiter ausgestatteten Version der *Trinity*-Bibliothek.

Außer der bereits erwähnten Modifikationen habe ich die CPNMouse-Bibliothek zusätzlich mit einem bereits im Bugtracker des Sourceforge-Projekts zur Verfügung stehenden Hotfix<sup>16</sup> ausgestattet, das einige

16 Hotfix for Bug # 1430536 ([http://sourceforge.net/tracker/index.php?func=detail&aid=1430580&group\\_id=92823&atid=602116](http://sourceforge.net/tracker/index.php?func=detail&aid=1430580&group_id=92823&atid=602116))

Bugs in der Low-Level-API behebt.

### 3.1.5 Modifikation des bestehenden Frameworks

Das bestehende Framework musste in allen Bereichen, in denen die Eingabe durch Zeigegeräte eine Rolle spielt, an die neuen Bedingungen angepasst werden. Um die notwendigen Veränderungen möglichst einfach und übersichtlich zu halten, versuchte ich die Struktur des Fenstersystems so wenig wie möglich anzutasten. Die größten Modifikationen mussten daher an der Managementklasse *CTUserInterface* vorgenommen werden; die Fensterbasisklasse *CTWindow* wurde nur geringfügig angepasst. Dies ersparte weitreichende und aufwändige Modifikationen an den von *CTWindow* ableitenden Subklassen.

#### 3.1.5.1 Identifikation und Verwaltung von mehreren Zeigegeräten

Bevor ich näher auf die programmiertechnischen Veränderungen innerhalb der *CTUserInterface* Klasse eingehe, möchte ich die für die Anwendung notwendig gewordene Identifikation und Verwaltung mehrerer Zeigegeräte notwendige Infrastruktur erläutern. Da die Anwendung von vorn herein für eine zweihändige Bedienung konzeptioniert war, werden die beiden Zeigegeräte auf oberster Ebene nicht über zahlenbasierte Indizes identifiziert (wie in *CPNMouse* gelöst), sondern über »Hände«. Intern verläuft die Identifikation natürlich weiterhin über Indizes, um die Kompatibilität zu *CPNMouse* zu gewährleisten. Welchen Index ein an das System angeschlossene Zeigegerät erhält, hängt von der dem unterliegenden *CPNMouse*-Treiber ab. Dieser verteilt die identifizierenden Indizes auf Basis der USB-Port-Identifikationsnummern<sup>17</sup>. Die Bezeichnung durch eine hierüber stehenden Abstraktionsebene, die »Hände«, ermöglicht es Zeigegeräte, unabhängig von ihrer Anschlussreihenfolge am System zu identifizieren. Der Datentyp *Hand* kann die beiden Werte *left* oder *right* annehmen. Die letztendliche Zuordnung einer *Hand* zu einem Zeigegerätindex erfolgt über eine Konfigurationsdatei.

#### 3.1.5.2 Modifikationen an User Interface und Mausnachrichten

Im Wesentlichen wurden in der Klasse *CTUserInterface* alle Attribute und Funktionen auf die Verarbeitung von zwei statt einer Eingabemöglichkeit durch Zeigegeräte umgestellt. Die folgende Tabelle zeigt die markantesten Unterschiede zwischen beiden Versionen auf:

---

17 Experimentell ermittelt

Einhändiges Interface	Zweihändiges Interface
Erfassung einer einzigen Mausposition	Erfassung zweier Mauspositionen
Registrierung von drei Maustasten einer Maus	Registrierung von sechs Maustasten (drei pro Maus)
Ermittlung einer Fensterliste der aktuell unter dem Mauszeiger befindlichen Fenster (MOUSEOVER)	Ermittlung zweier Fensterlisten der aktuell unter den Mauszeigern befindlichen Fenster (MOUSEOVER)
Ermittlung einer Fensterliste der von dem Mauszeiger gerade verlassenen Fenster (MOUSELEAVE)	Ermittlung zweier Fensterlisten der aktuell von den Mauszeigern verlassenen Fenster (MOUSELEAVE)

Um die Veränderungen innerhalb des Fenstersystems selbst so gering wie möglich zu halten, habe ich zunächst keine zusätzlichen Mausnachrichten<sup>18</sup> eingeführt. Stattdessen wurden die mit den Mausnachrichten verknüpften Zusatzinformationen erweitert. Statt wie vorher lediglich die aktuelle Mausposition (vgl. S. 7) mit einer Mausnachricht zu übermitteln, wurde nun zusätzlich die *Hand* der Maus, die für das Senden der jeweiligen Nachricht ausschlaggebend war, an das Fenstersystem gesendet. Dies bedeutet, dass alle von *CTWindow* abgeleiteten Subklassen (Schaltflächen, Bilder, Control- / Statusbars, etc.) weiterhin wie gewohnt funktionieren, nun allerdings auf Eingaben beider Zeigegeräte gleich reagieren. In der Praxis ist es also nun egal, ob mit der linken oder der rechten Maus beispielsweise eine Schaltfläche bedient wird. Fensterobjekte, die speziell für das zweihändige System ausgelegt werden sollen, können durch Auslesen des *Hand*-Parameters problemlos spezialisiert werden.

---

18 Bereits bestehende (für die Anwendung relevante) Nachrichten siehe S. 14 Abb. 8.

# 4 Die Studie

## 4.1 Methode

### 4.1.1 Rahmenbedingungen

Die Studie wurde mit Hilfe der Testanwendung an einem PC-Notebook mit 15"-TFT-Display (1400 x 1050 Pixel Auflösung) durchgeführt. Als Eingabegeräte wurden zwei Microsoft IntelliMouse Optical USB-Mäuse verwendet, welche sowohl für Links- als auch für Rechtshänder geeignet sind. Eine Maus wurde links neben dem Notebook platziert, die andere rechts. In der Windows-Systemsteuerung des PCs wurde als Geschwindigkeit für die Mauszeiger der viertlangsamste Wert ausgewählt. Um möglichst gleichbleibende Testbedingungen zu schaffen, wurden stets zwei gleiche Mauspads als Unterlage für die Mäuse wiederverwendet.

### 4.1.2 Aufgabe

#### 4.1.2.1 Aufgabenstellung

Die Probanden sollten folgende Aufgabe für beide Interaktionsmodelle durchführen:

»Auf dem Bildschirm sind drei graue Referenzobjekte zu sehen, die jeweils eine andere Form repräsentieren. Weiterhin befinden sich drei farbige, Objekte auf der Arbeitsfläche. Transformieren Sie die farbigen Formen so, dass die ihre Markierungspunkte und die Markierungspunkte der Referenzobjekte möglichst präzise auf den Referenzobjekten liegen. Führen Sie die Aufgabe drei mal hintereinander aus.«

#### 4.1.2.2 Anforderungen an die Aufgabenstellung

Das Design des Aufgabengegenstands sowie die Aufgabenstellung selbst wurden aufgrund folgender Gesichtspunkte festgelegt:

- **Verständlichkeit**  
Keiner der Probanden, auch solche, die wenig Erfahrung im Umgang mit PCs haben, sollte mit dem Verständnis der Aufgabenstellung und des Gegenstands der Aufgabe überfordert werden. Dies war

auch deshalb wichtig, da die Probanden einen Fragebogen für die qualitative Evaluation der Studie möglichst selbständig ausfüllen können sollten.

- **Eindeutigkeit**

Die Aufgabenstellung sollte möglichst eindeutig definierbar sein. Darüber hinaus sollte der Gegenstand der Aufgabe selbst möglichst eindeutig gestaltet sein, so dass die Probanden durch keinerlei Doppeldeutigkeiten oder Unregelmäßigkeiten beeinflusst werden.

- **Messbarkeit**

Die Aufgabe sollte möglichst gut messbar, also quantitativ evaluierbar sein.

- **Vergleichbarkeit**

Die Aufgabe sollte eine möglichst gute Vergleichbarkeit beider Interaktionsmodelle ermöglichen.

- **Ähnlichkeit mit bekannten Anwendungsszenarien**

Das Szenario der Testaufgabe sollte möglichst auf eine reale, alltägliche Anwendung übertragbar sein. Die Aufgabe wurde so gewählt, dass sie im weitesten Sinne der des Layouting entspricht. Dabei mussten hier die Möglichkeiten der Anwendung stark eingeschränkt werden, da sonst andere Kriterien wie Messbarkeit und Eindeutigkeit beeinflusst worden wären.

#### 4.1.3 Prozedur

Vor jedem Test wurden der Testperson beide Interaktionsmodelle kurz erklärt. Dabei wurden keine Tipps zu den speziellen Charakteristika des jeweiligen Interaktionsmodells und den damit verbunden Vor- oder Nachteilen bei der Benutzung gegeben. Jedoch wurde aufgrund der Erwartungen an die Studie (siehe 4.1.5) deutlich erklärt, dass nur die linke Maustaste beider Mäuse eine Funktion besitzt, und dass die Maus der nichtdominanten Hand im asymmetrischen Modell überhaupt keine Klickfunktion inne hat. Die in 4.2.1.1 aufgezeigte Aufgabenstellung wurde mündlich vorgetragen.

Der Test begann stets mit der Durchführung der Aufgabe im symmetrischen Modell, darauf folgte die gleiche Aufgabe im asymmetrischen Modell. Für jedes Modell sollten die Probanden die Aufgabe jeweils drei Mal absolvieren. Dies diente dazu, festzustellen, inwiefern die Eingewöhnung in den Manipulationsablauf bei den Interaktionsmodellen zu einer Verbesserung der Testergebnisse führte. Ein Zeitlimit für die Durchführung der Gesamtaufgabe wurde nicht vorgegeben. Somit konnte im Nachhinein festgestellt werden, wie viel Zeit die Probanden für die Absolvierung der Aufgabendurchläufe benötigten und inwiefern sich ihre Geschwindigkeit durch Eingewöhnung verbesserte.

#### 4.1.4 Probanden

Zehn Personen im Alter zwischen 21 und 64 Jahren nahmen an der Studie teil (drei Frauen, sieben Männer). Alle der Probanden gaben an rechtshändig zu sein. Alle Probanden benutzten PCs in ihrem Arbeitsalltag und hatten vorherige Erfahrung im Umgang mit Computermäusen.

#### 4.1.5 Erwartungen

Aufgrund der Erfahrungen, die ich selbst während der Entwicklung und dem Ausprobieren der Testanwendung gesammelt hatte, war es zu erwarten, dass insbesondere eine Entscheidung, die ich bei dem Design der Bedienabläufe getroffen hatte, kontroverse Reaktionen bei den Probanden hervorrufen würde: die Belegung der Maustasten. In vorherigen Tests hatte sich herausgestellt, dass einige Nutzer bei der linken Maus instinktiv die rechte, unter dem Zeigefinger liegende Taste als Auslöser nutzen. Andere wählten wiederum von sich aus die linke Taste (siehe 4.2.1.4, 4.2.1.5 bzw. 4.2.2.4).

Abgesehen von diesem Kriterium, das im Vorfeld nur schwer einschätzbar war, waren meine Erwartungen an die Ergebnisse der Studie ansonsten relativ klar. Ich rechnete damit, dass in beiden Modellen über die drei jeweils ausgeführten Durchläufe hinweg eine deutliche Verbesserung sowohl in den Handling- als auch Zeitwerten auftreten würde. Darüber hinaus erwartete ich, dass das symmetrische gegenüber dem asymmetrischen Modell weniger Zeit und signifikant weniger Klicks, dafür jedoch auch eine etwas geringere Präzision zum Ergebnis haben sollte.

Was die Einschätzungen der Probanden anbetrifft, erwartete ich, dass das symmetrische dem asymmetrischen Modell generell vorgezogen würde, da ich es für intuitiver und leichter handhabbar hielt. Darüber hinaus erwartete ich, dass die meisten der Testnutzer keine großen Eingewöhnungsprobleme mit der Toolglass-Technik haben würden, da dies eines der Ergebnisse von Kabbash, Buxton und Sellen [Kabbash, P. (1994)] war.

## 4.2 Evaluation

### 4.2.1 Quantitative Evaluation

#### 4.2.1.1 Messung

Zur Messung der Benutzereingaben wurde ein in die Testanwendung integriertes Logging-System verwendet. Die Art der Daten und ihrer Protokollierung möchte ich im Folgenden kurz erläutern:

- **Präzisionsdaten**

Bei Bewegung einer der Mäuse errechnet das System alle 100 Millisekunden<sup>19</sup>, wie präzise sich die auf der Arbeitsfläche liegenden Objekte mit den Referenzobjekten decken. Dazu werden die Bounding-Boxes der Objekte und der jeweils zu ihnen gehörenden Referenzobjekte verwendet. Das System bestimmt beispielsweise die mittlere Entfernung zwischen den Punkten des Rechtecks und denen des Referenzrechtecks in Pixeln. Je kleiner der so errechnete Wert wird, desto präziser wurde das Rechteck auf dem Referenzrechteck abgelegt. Je größer er wird, desto ungenauer hat der Benutzer das Objekt platziert.

- **Mauspositionen**

Bei jeder Mausbewegung wird die Position (x- und y-Wert) des bewegten Mauszeigers auf der Arbeitsfläche protokolliert.

- **Mausklicks**

Jedes Drücken oder Loslassen der Maustasten wird protokolliert. In die Erfassung werden alle Maustasten beider Mäuse mit einbezogen, obwohl in beiden Interaktionsmodellen nur die linke Maustaste mit einer Funktion belegt ist.

- **Umgebungseinstellungen (aktuelle Modi)**

Um die jeweiligen Umgebungseinstellungen zu erfassen, wird jedes Mal wenn ein Modus geändert wird, ein Eintrag im Protokoll hinterlegt (Hin- und Herschalten zwischen Links- oder Rechtshändermodus und symmetrischem und asymmetrischem Modell).

Alle erfassten Daten wurden durch das System mit einem millisekundengenauen Zeitindex, der bei jeder

---

<sup>19</sup> Um die Prozessorauslastung gering zu halten, wurde die Berechnung der Präzisionsdaten auf dieses Intervall beschränkt.

Modusänderung auf Null zurückgesetzt wurde, versehen. Dadurch ließ es sich nach Durchführung der Tests leicht nachvollziehen, wann eine Operation zu einer bestimmten Konsequenz geführt hatte.

#### 4.2.1.2 Vergleichskriterien

- **Benötigte Gesamtzeit**  
Angabe der benötigten Gesamtzeit in Sekunden zur Absolvierung eines Aufgabendurchlaufs.
- **Präzision des Endergebnisses**  
Bewertung auf Basis einer Skala von 1 bis 10, wobei 10 einem Präzisionswert von 0,0 Pixeln und 1 einem Wert von 100,0 Pixeln entspricht (linear interpoliert).
- **Präzisionsentwicklung**  
Bewertung auf Basis einer Skala von 1 bis 10, wobei 10 einer ständig (ohne Unterbrechung) zunehmenden Präzision und 1 einer zunehmenden Präzision mit zwanzig Unterbrechungen (vorübergehend schlechter werdenden Präzision) entspricht.
- **Gesamtanzahl von Klicks mit der linken Maustaste der linken Maus**  
Die Anzahl der Klicks mit der linken Maustaste der linken Maus nach der vollständigen Absolvierung.
- **Gesamtanzahl von Klicks mit der rechten Maustaste der linken Maus**  
Die Anzahl der Klicks mit der rechten Maustaste der linken Maus nach der vollständigen Absolvierung.
- **Gesamtanzahl von Klicks mit der linken Maustaste der rechten Maus**  
Die Anzahl der Klicks mit der linken Maustaste der rechten Maus nach der vollständigen Absolvierung.
- **Verhältnis der Bewegungshäufigkeit beider Mäuse**  
Bewertung auf Basis des Quotienten 'Linke Maus' : 'Rechte Maus'; eins zu zwei (1 : 2) bedeutet also, dass die linke Maus halb so oft benutzt wurde wie die rechte.
- **Verhältnis der Mauswege**  
Bewertung auf Basis des Quotienten 'Linke Maus' : 'Rechte Maus'; eins zu zwei (1 : 2) bedeutet also, dass die linke Maus halb so weit bewegt wurde wie die rechte.

### 4.2.1.3 Auswertung der Messdaten

Wie in 3.1.5 angesprochen, wurde für Konvertierung und Auswertung der Messdaten ein externes Kommandozeilenprogramm entwickelt, welches die aufgezeichnet Binärdaten des Protokolls filtert und in CSV-Dateien umwandelt. Mit Hilfe von Tabellenkalkulationsprogrammen ließen sich aus diesen Dateien Tabellen, Graphen und Diagramme erzeugen, die die Ergebnisse der Datenerhebungen visualisierbar machten. Diese lieferten wichtige Anhaltspunkte über die Wahl der Kriterien und zeigten exemplarisch das Verhalten der Probanden bei der Nutzung der Testanwendung auf. Die entstandenen Tabellen wurden manuell nachbearbeitet, um unbrauchbare und leere Aufgabendurchläufe<sup>20</sup> zu eliminieren. Das Auswertungsprogramm wurde mit Routinen zur Umrechnung in die in 4.2.1.2 aufgeführten Vergleichskriterien ausgestattet und lieferte so für jeden Durchlauf aus den nachbearbeiteten CSV-Dateien die entgültigen Ergebniswerte. Aus diesen Ergebniswerten wurden Durchschnittswerte errechnet. Außerdem wurden die jeweils besten und schlechtesten Werte pro Aufgabendurchlauf und Kriterium jeweils eines Probanden ermittelt.

### 4.2.1.4 Ergebnisse

Im Folgenden stelle ich die Ergebnisse der quantitativen Evaluation tabellarisch dar. Alle Werte sind auf zwei Nachkommastellen gerundet. Da die beiden letzten Kriterien (*Differenz der Bewegungshäufigkeit beider Mäuse* und *Differenz der Mauswege*) aufgrund fehlender Beurteilungsmaßstäbe nicht auf beste oder schlechteste Werte hin überprüfbar sind, wurden sie in den Tabellen 2 und 3 weggelassen.

Tabelle 1: Durchschnittswerte:

Modell	symmetrisch				asymmetrisch			
	1	2	3	G*	1	2	3	G*
Durchlauf								
Kriterien:								
Benötigte Gesamtzeit	152,26	89,04	81,1	<b>107,46</b>	147,87	111,6	130,14	<b>138,93</b>
Präzision des Endergebnisses	8,18	9,27	9,23	<b>8,92</b>	9,19	9,61	8,13	<b>9,21</b>
Präzisionsentwicklung	3,47	4,76	5,67	<b>4,63</b>	4,09	4,23	3,88	<b>4,09</b>
Klicks mit linker Taste der linken Maus	18,45	24,73	16,72	<b>14,64</b>	11,45	6,55	6,36	<b>8</b>
Klicks mit rechter Taste der linken Maus	2,9	0	0	<b>0,9</b>	0	0	0	<b>0</b>
Klicks mit linker Taste der rechten Maus	46,27	30,73	25,82	<b>34</b>	69	51,09	49,81	<b>56,63</b>
Differenz d. Bewegungshäufigk. beider M	1:1,67	1:1,64	1:1,63	<b>1:1,65</b>	1:2,95	1:2,71	1:3,14	<b>1:2,92</b>
Differenz der Mauswege	1:1,24	1:1,18	1:1,02	<b>1:1,11</b>	1:1,64	1:1,34	1:1,54	<b>1:1,51</b>

<sup>20</sup> Solche wurden korrekt wiederholt, jedoch trotzdem in den Logdateien gespeichert.

Tabelle 2: Beste Werte (pro Aufgabendurchlauf und Kriterium jeweils eines Probanden):

Modell	symmetrisch				asymmetrisch				
	1	2	3	G*	1	2	3	G*	
Durchlauf									
Kriterien:									
Benötigte Gesamtzeit	61,72	59,51	41,38	<b>64,06</b>	69,92	41,28	73,8	<b>55,6</b>	
Präzision des Endergebnisses	9,88	9,87	9,92	<b>9,87</b>	9,9	9,89	9,89	<b>9,87</b>	
Präzisionsentwicklung	6,33	6,83	7,33	<b>5,94</b>	6	7	6,33	<b>6,38</b>	
Klicks mit linker Taste der linken Maus	10	6	6	<b>8</b>	0	0	0	<b>0</b>	
Klicks mit rechter Taste der linken Maus	0	0	0	<b>0</b>	0	0	0	<b>0</b>	
Klicks mit linker Taste der rechten Maus	9	8	8	<b>9</b>	32	22	29	<b>25</b>	

Tabelle 3: Schlechteste Werte (pro Aufgabendurchlauf und Kriterium jeweils eines Probanden):

Modell	symmetrisch				asymmetrisch			
	1	2	3	G*	1	2	3	G*
Durchlauf								
Kriterien:								
Benötigte Gesamtzeit	328,74	151,99	134,78	<b>176,56</b>	351,38	208,08	274,51	<b>273,36</b>
Präzision des Endergebnisses	5,26	5,53	5,56	<b>6,9</b>	5,49	9,07	5,45	<b>5,49</b>
Präzisionsentwicklung	1	2,67	4	<b>3,33</b>	2,5	1,67	1,83	<b>2,22</b>
Klicks mit linker Taste der linken Maus	40	30	36	<b>28</b>	18	24	22	<b>21</b>
Klicks mit rechter Taste der linken Maus	22	0	0	<b>7</b>	0	0	2	<b>0</b>
Klicks mit linker Taste der rechten Maus	107	92	71	<b>77</b>	171	92	103	<b>118</b>

\*G = GesamtDurchschnitt pro Modell

#### 4.1.2.5 Analyse

##### Benötigte Zeit

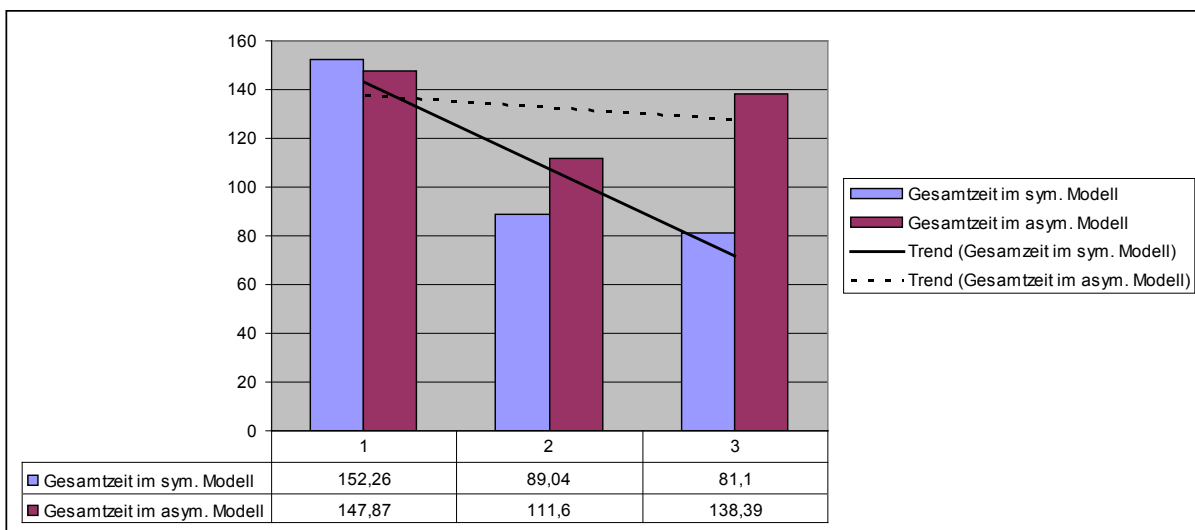


Diagramm 1: Entwicklung der durchschnittlich benötigten Zeit (Y-Achse) pro Aufgabendurchlauf (X-Achse) und Modell

Diagramm 1 zeigt den Verlauf der durchschnittlich benötigten Zeit aller Probanden pro Aufgabendurchlauf und Interaktionsmodell. Dabei fällt auf, dass sich die benötigte Zeit im symmetrischen Interaktionsmodell kontinuierlich verringert. Bereits beim zweiten Durchlauf halbierte sie sich durchschnittlich nahezu. Im Gegensatz dazu zeigt die Entwicklung im asymmetrischen Modell, dass sich die benötigte Zeit zwar auch hier verringert, jedoch scheinen viele Probanden für den dritten und letzten Durchlauf wieder mehr Zeit benötigt zu haben. Somit bleibt die Zeit im Gesamtdurchschnitt eher stabil. Die Ergebnisse decken sich hier also nur im symmetrischen Modell mit den Erwartungen.

### Präzision des Endergebnisses

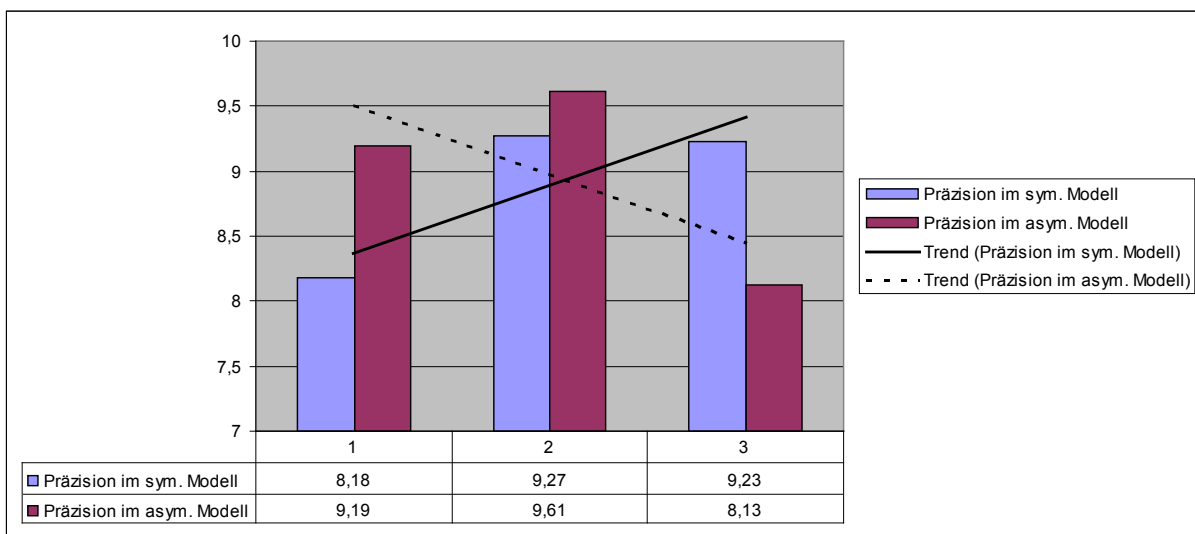


Diagramm 2: Durchschnittlich Präzision (Y-Achse) des Endergebnisses pro Aufgabendurchlauf (X-Achse) und Modell

Wie Diagramm 2 aufzeigt, gibt es auch im Hinblick auf die Präzision des Endergebnisses in den jeweiligen Aufgabendurchläufen eine unerwartete Auswertung des asymmetrischen Modells. Hier wird die Präzision tendenziell eher etwas geringer, wobei sie beim symmetrischen Modell, wie prognostiziert, leicht zunimmt. Alle Präzisionswerte befinden sich jedoch konstant auf einem recht hohen Niveau. Wie aus Tabelle 1 ersichtlich wird, trifft die Erwartung zu, dass das asymmetrische Modell insgesamt präzisere Ergebnisse hervorbringt. So liegt der Gesamtdurchschnitt des Präzisionswertes beim symmetrischen Modell mit 8,92 Punkten fast 0,3 Punkte unter dem des asymmetrischen Modells mit 9,21 Punkten.

## Präzisionsentwicklung

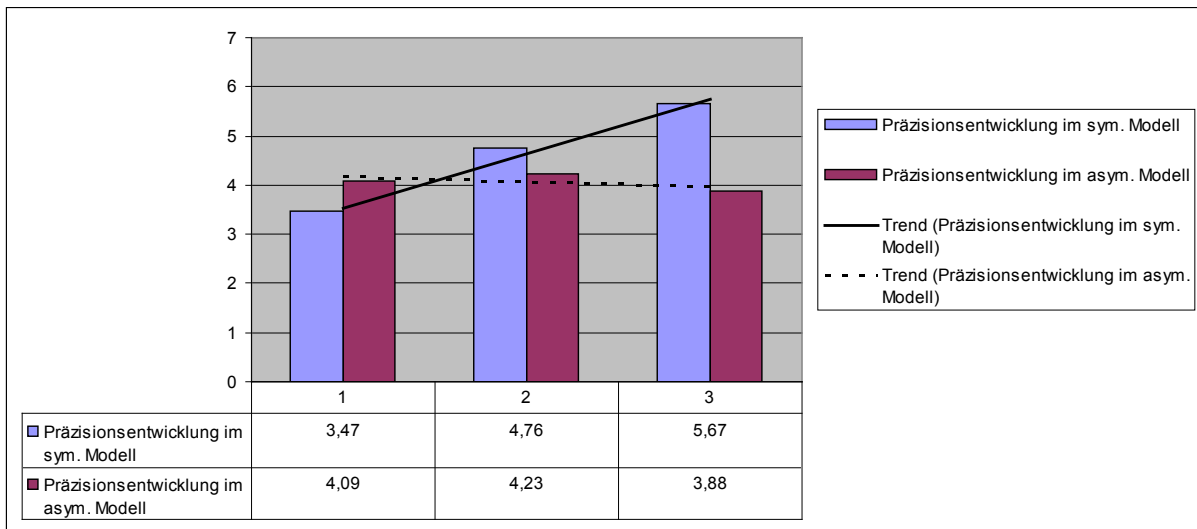


Diagramm 3: Entwicklung der Präzision (Y-Achse) pro Aufgabendurchlauf (X-Achse) und Modell

Aus Diagramm 3 wird ersichtlich, dass sich die Präzisionsentwicklung im symmetrischen Modell kontinuierlich verbessert, während sie im asymmetrischen Modell nahezu konstant bleibt. Dies deutet darauf hin, dass die Probanden im symmetrischen Modell durch Eingewöhnung eine effizienter werdende Motorik und Steuerungsfähigkeit erreicht haben. Im asymmetrischen Modell hingegen blieb die Herangehensweise und damit die Anzahl an Verschlechterungen der Lage der Objekte während der Versuche gleich.

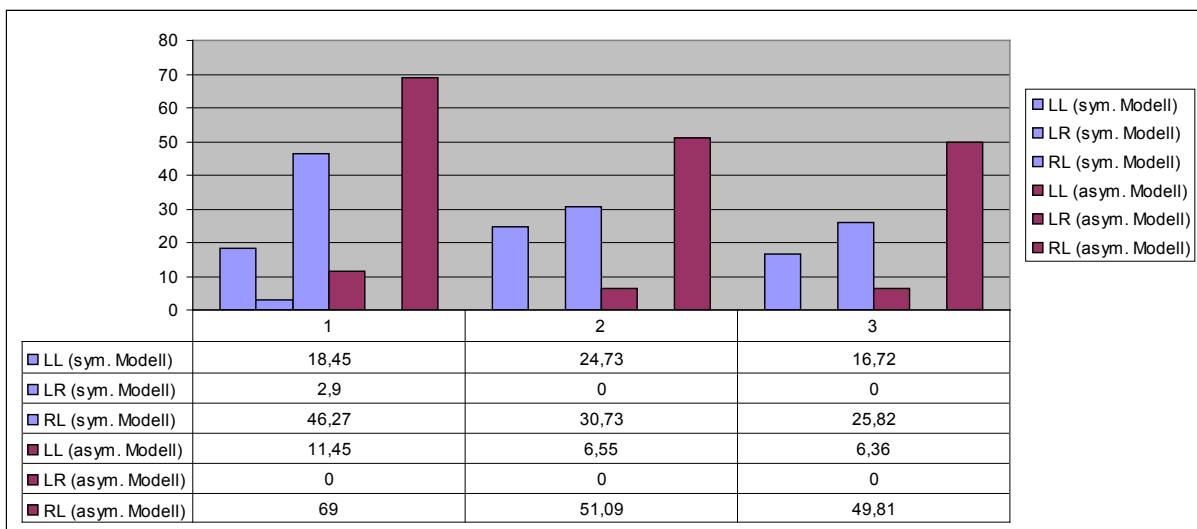


Diagramm 4: Durchschnittliche Anzahl von Mausclicks (Y-Achse) pro Aufgabendurchlauf (X-Achse) und Modell; nur Modelle farblich unterschieden; Reihenfolge der Legende trifft auch auf die Diagrammdarstellung zu.

Diagramm 4 zeigt alle ausgewerteten Durchschnittlichen Anzahlen von Mausclicks. Dabei bedeutet LL: linke Maus, linke Taste; LR: linke Maus, rechte Taste; RL: rechte Maus, rechte Taste. Letztere ist die im Alltag

ständig benutzte Maustaste. Dies zeigt sich deutlich in den ermittelten Werten. Die linke Taste der rechten Maus ist mit durchschnittlich 30 Klicks im symmetrischen bzw. gut 56 Klicks im asymmetrischen Modell die erste Wahl der Probanden. Im symmetrischen Modell erkennt man jedoch eine Angleichung der Klickhäufigkeit der linken und der rechten Maus mit wachsender Eingewöhnungszeit. Interessant an den Durchschnittswerten für die Mausklicks ist jedoch, dass viele Probanden auch im asymmetrischen Modell mit der linken Maus geklickt haben, obwohl ihnen vorher ausdrücklich gesagt wurde, dass die linke Maus hier keine Tastenfunktion hat. Des Weiteren zeigen die Werte für die rechte Maustaste der linken Maus, dass einige Probanden am Anfang versuchten, diese zu betätigen, was wie bereits in 4.1.5 angesprochen den Erwartungen entsprach.

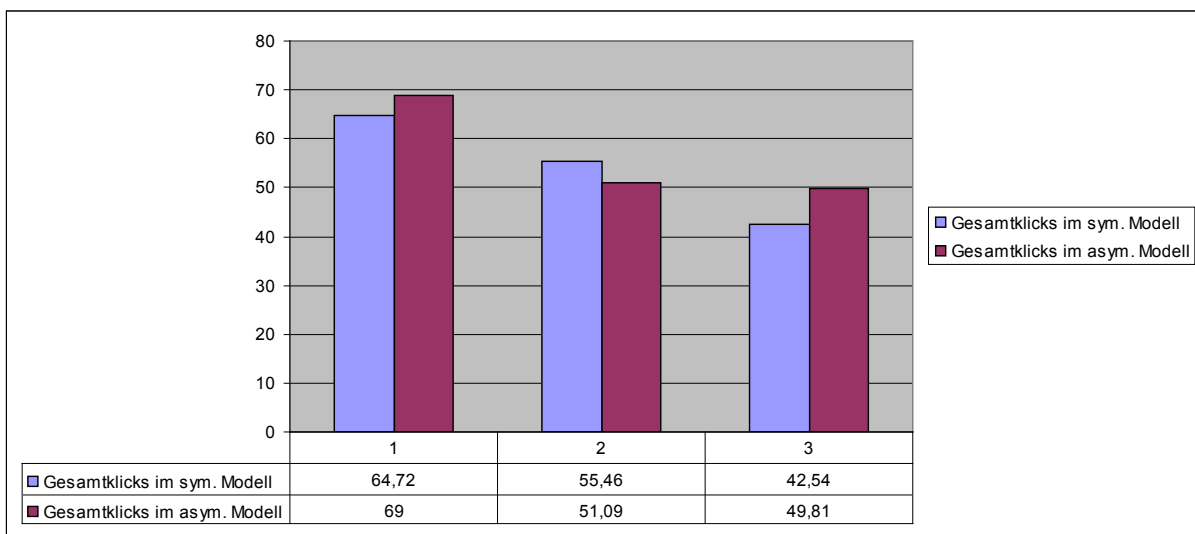


Diagramm 5: Durchschnittliche Gesamtanzahl von Mausklicks (Y-Achse) pro Aufgabendurchlauf (X-Achse) und Modell

Insgesamt nahm die Anzahl der Klicks generell mit steigender Eingewöhnung ab, was ebenfalls die Erwartungen erfüllt. Dies ist besonders gut in Diagramm 5 zu erkennen. Des Weiteren sieht man, dass die Gesamtanzahl der Klicks in beiden Modellen immer ungefähr gleich ist, wobei hierbei das symmetrische Modell stärker dazu tendiert mit wachsender Eingewöhnung effizienter zu werden. Einen signifikanten Unterschied in der Gesamtklickanzahl gibt es zwischen beiden Modellen entgegen der Erwartungen nicht.

### Bewegungshäufigkeit

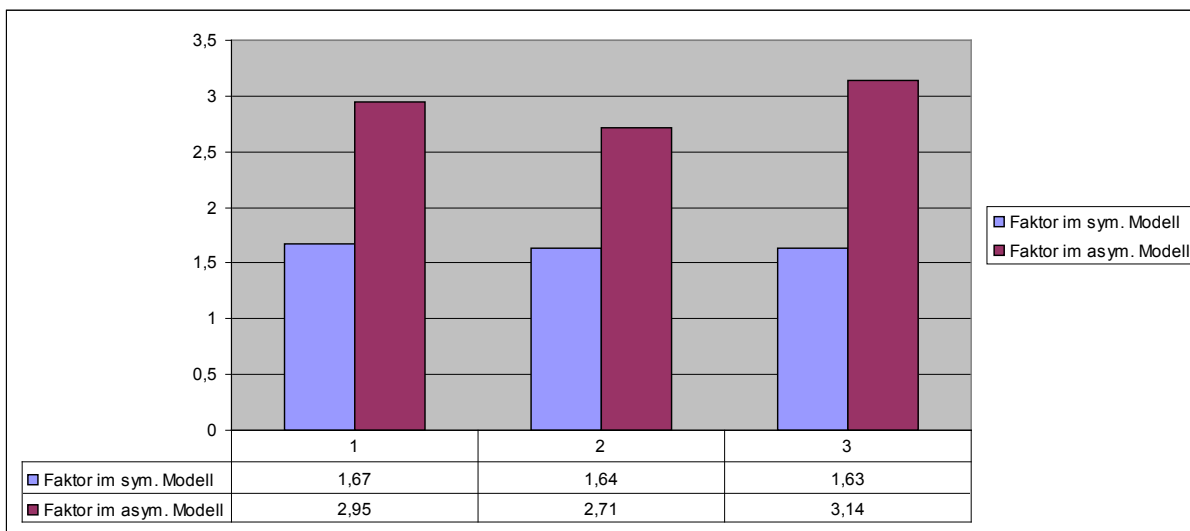


Diagramm 6: Durchschnittlicher Faktor des Verhältnisses der Bewegungshäufigkeit der rechten Maus gegenüber der linken Maus (Y-Achse) pro Aufgabendurchlauf (X-Achse) und Modell

Diagramm 6 zeigt den Nenner des Quotienten zur Bewertung des Verhältnisses der Bewegungshäufigkeit zwischen beiden Mäusen. Deutlich zu erkennen ist, dass die rechte Maus ungeachtet vom Modell im Durchschnitt häufiger bewegt wurde. Im asymmetrischen Modell wurde die rechte Maus insbesondere bevorzugt, was jedoch durch die Natur des Bedienablaufs im Modell selbst begründet ist. Interessant ist, dass es bei diesem Kriterium kaum eine Veränderung durch Eingewöhnung gibt.

### Mauswege

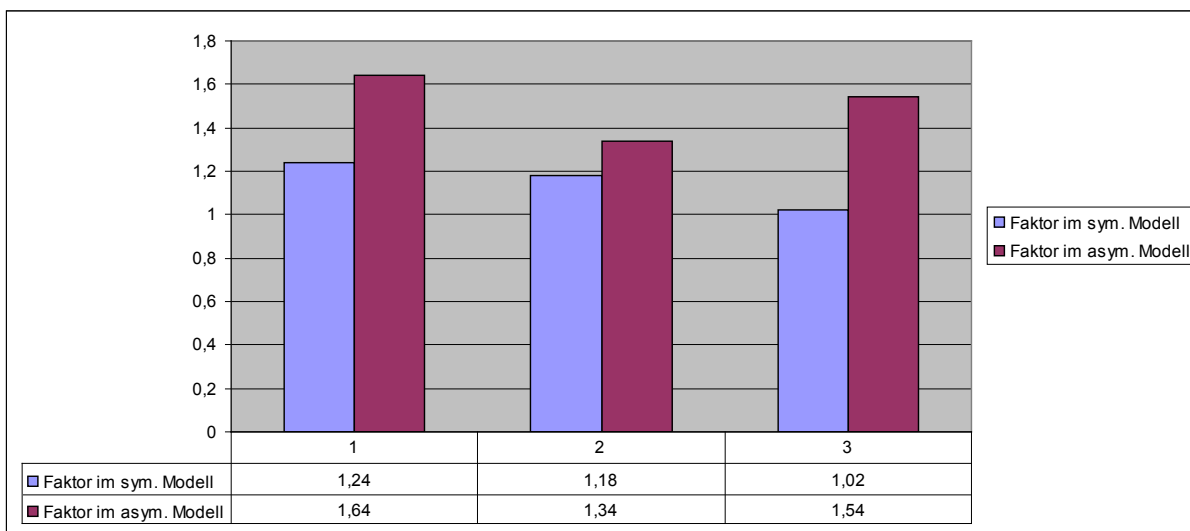


Diagramm 7: Durchschnittlicher Faktor des Verhältnisses der Mauswege der rechten Maus gegenüber der linken Maus (Y-Achse) pro Aufgabendurchlauf (X-Achse) und Modell

Diagramm 7 zeigt den Nenner des Quotienten zur Bewertung des Verhältnisses der Mauswege zwischen

beiden Mäusen. Bei der Analyse zeigt sich, dass die Verhältnisse hier im Gegensatz zur Bewegungshäufigkeit ausgeglichener sind. Im Mittel ist das Verhältnis zwischen den Mauswegen der linken und der rechten Maus im symmetrischen Modell 1:1,11, im asymmetrischen 1:1,51. Bei der Bewegungshäufigkeit ist die Differenz deutlich größer: 1:1,65 beim symmetrischen und 1:2,92 im asymmetrischen Modell. Die linke Maus wird also deutlich seltener bewegt, legt dabei aber – je nach Modell – fast genau so viel Strecke zurück. Diagramm 7 zeigt außerdem eine Ausgleichung des Verhältnisses mit zunehmender Erfahrung beim symmetrischen Modell. Beim asymmetrischen Modell hingegen bleibt das Verhältnis relativ konstant.

#### 4.2.1.5 Bemerkung zur Analyse

Aufgrund der Tatsache, dass alle Teilnehmer der Studie rechtshändig waren, wurde auf eine Verallgemeinerung für die nichtdominante Hand in der Auswertung und Analyse verzichtet. Die linke Maus ist hier immer als Eingabegerät der nichtdominanten Hand zu verstehen.

### 4.2.2 Qualitative Evaluation

#### 4.2.2.1 Datenerhebung

Zur qualitativen Evaluation der Studie wurde von den Probanden nach Beendung des Versuchs ein Fragebogen<sup>21</sup> ausgefüllt. Auf dem Bogen wurden Fragen zur Vorerfahrung, Eingewöhnung und Beurteilung gestellt. Die Antworten konnten die Probanden auf einer Skala von sechs möglichen Markierungen gewichten. Für Beurteilungsfragen zwischen beiden Interaktionsmodellen wurde zusätzlich die Beurteilung »gleich« angeboten. Für alle Fragen gab es die Möglichkeit mit »keine Angabe (k.A.)« zu antworten. Abschließend konnten die Probanden Kommentare niederschreiben. Zusätzlich wurden folgende Personenbezogene Daten abgefragt: Alter, Geschlecht, Rechtshänder / Linkshänder. Alle Fragebogen wurden fortlaufend durchnummeriert, um sie gegebenenfalls den Logfiles der quantitativen Evaluation zuordnen zu können.

#### 4.2.2.2 Auswertung

Zur Auswertung der Fragebögen wurde von allen gestellten Fragen Durchschnittswerte der angekreuzten Antworten ermittelt. Falls jemand bei Beurteilungsfragen zwischen beiden Modellen mit »gleich« geantwortet hatte, wurde dies gesondert behandelt. Keiner der Probanden antwortete jemals mit »keine Angabe (k.A.)«, daher wurde diese Antwortmöglichkeit nicht in die Auswertung mit einbezogen. Darüber hinaus

---

21 Originalfragebogen siehe Appenix I (S.40)

wurde das Minimum, das Maximum sowie der Durchschnitt des Alters der Probanden errechnet sowie die anderen personenbezogenen Daten ausgewertet. Außerdem wurden alle Kommentare analysiert. Interessante und oft wiederkehrende Kommentare werden in 4.2.2.4 gesondert erwähnt.

### 4.2.2.3 Ergebnisse

#### Probanden

Die Probanden waren zwischen 21 und 64 Jahren alt. Das Durchschnittsalter betrug 38,2 Jahre. Drei waren weiblich, die anderen sieben männlich. Alle waren rechtshändig, wobei einer der Probanden angab, beidhändig zu sein, jedoch mit der rechten Hand zu schreiben. Dieser Proband wurde ebenfalls als Rechtshänder behandelt.

#### Ergebnisse der Fragebogenauswertung

Frage	Antwort 1	Gewichtung	Antwort 2	»gleich«
Wie oft arbeiten Sie im Alltag am PC?	sehr oft	<input checked="" type="checkbox"/> □□□□□	sehr selten	
Wie viel Erfahrung haben Sie mit herkömmlichen Bildbearbeitungs- oder Layouting-Programmen?	sehr viel	□□□ <input checked="" type="checkbox"/> □□	keine	
Haben Sie praktische Erfahrung mit ähnlichen zweihändigen Bedienungsformen?	viel	□□□□□ <input checked="" type="checkbox"/>	keine	
Wie lange haben Sie gebraucht, um das Konzept der zweihändigen Bedienung zu verstehen?	kurz	□□ <input checked="" type="checkbox"/> □□□	lange	
War es für Sie schwierig sich an die Bedienung mittels zwei Mäusen zu gewöhnen?	sehr	□□□□ <input checked="" type="checkbox"/> □	gar nicht	
Fanden Sie die Bedienung mittels zwei Mäusen angenehm?	sehr	□□ <input checked="" type="checkbox"/> □□□	gar nicht	
Welches Bedienungsmodell sagte Ihnen mehr zu? (A = symmetrisches; B= asymmetrisches Modell)	A	□ <input checked="" type="checkbox"/> □□□□	B	33,3%
Welches Bedienungsmodell war für Sie intuitiver zu handhaben?	A	□□ <input checked="" type="checkbox"/> □□□	B	-
Hatten Sie generell das Gefühl mit der zweihändigen Bedienung effizient zu arbeiten?	sehr	□□ <input checked="" type="checkbox"/> □□□	gar nicht	
Bei welchem Bedienmodell hatten Sie das Gefühl effizienter zu arbeiten?	A	□□□ <input checked="" type="checkbox"/> □□	B	-
Könnten Sie sich vorstellen eine zweihändige Interaktion in Ihrem Arbeitsalltag am PC anzuwenden?	ja	□□ <input checked="" type="checkbox"/> □□□	nein	

#### Zusammenfassung

Alle Probanden arbeiteten in ihrem Alltag sehr oft an PCs und ca. die Hälfte von ihnen hatte Erfahrungen mit herkömmlichen Bildbearbeitungs- oder Layouting-Programmen. Keiner der Probanden hatte im Vorfeld

schon einmal ein zweihändiges Bedienmodell benutzt. Die Zeit, die die Probanden zur Eingewöhnung in die zweihändige Interaktion benötigten, schätzten sie in der Regel selbst als recht lang ein. Im Durchschnitt hält sich diese Einschätzung im Mittelfeld zwischen kurz und lang. Sieht man sich die Werte für die benötigte Zeit der Probanden pro Aufgabendurchlauf an, die im Schnitt bei rund zwei Minuten lag (2:02 min.), so relativiert sich diese Selbsteinschätzung jedoch in der objektiven Betrachtung. Die meisten Probanden empfanden es als leicht, sich an die Bedienung mit zwei Mäusen zu gewöhnen. Die Ergebnisse aus 4.2.1.4 unterstützen diese Einschätzung. Sie zeigen, dass die Resultate sich generell im Laufe der Eingewöhnung verbessert haben. Es gab einige Probanden, die die zweihändige Bedienung generell angenehm fanden, im Mittel ist die Einschätzung für dieses Kriterium jedoch eher neutral ausgefallen. Generell bekam das symmetrische Modell den Vorzug wenn es um Gefallen und Intuitivität ging. Die Frage, welches beider Modelle ein effizienteres Arbeiten ermögliche, entschied sich jedoch leicht zu Gunsten des asymmetrischen Bedienansatzes. Schließlich war die Meinung über die Einsetzbarkeit von zweihändigen Interaktionen im Arbeitsalltag eher zweigeteilt. Im Schnitt stand hier ein »eher ja« als Antwort.

#### 4.2.2.4 Besonderheiten und Kommentare

- Zwei der Probanden nutzten in ihrem normalen Arbeitsalltag mit dem PC ausschließlich Touchpads und gaben daher an, weniger Praxis im Umgang mit Mäusen zu haben.
- Drei Probanden gaben an, besondere Probleme mit der Eingewöhnung in die Benutzung der Toolglass-Technik gehabt zu haben. Zwei der drei äußerten, dass die Rotation mit Hilfe des Toolglass unvorteilhaft sei, da es intuitiver wäre, wenn diese durch eine Kreisbewegung anstatt durch Hoch- und Runterbewegen der Maus gelöst worden wäre.
- Vier der Probanden gaben an, dass sie Probleme damit hatten, die linke Maustaste der linken Maus zum Klicken zu verwenden. Sie hätten die Belegung der rechten Taste für die linke Maus als intuitiver empfunden.

### 4.3 Diskussion und Beurteilung

Im Folgenden möchte ich die Ergebnisse der Auswertungen im Hinblick auf das gesetzte Ziel der Studie diskutieren. Ziel der Studie war es, beide Interaktionsmodelle auf Akzeptanz, Präzision, Effizienz, Intuitivität und Integration der nichtdominanten Maus in den Bedienablauf zu vergleichen. Ich gehe nun detailliert auf diese Vergleichskriterien ein und zeige dabei auf, inwiefern die Auslegung der quantitativen und qualitativen Evaluation diese klären können.

### *Akzeptanz*

Zur Bestimmung der Akzeptanz ließ sich lediglich die qualitative Evaluation heranziehen, da die Messdaten der Testanwendung selbst keine Aussage hierüber machen können. Von Belang waren hier insbesondere die Fragen 4 (Wie lange haben Sie gebraucht, um das Konzept der zweihändigen Bedienung zu verstehen?), 5 (War es für Sie schwierig, sich an die Bedienung mittels zwei Mäusen zu gewöhnen?), 6 (Fanden Sie die Bedienung mittels zwei Mäusen angenehm?) und 11 (Könnten Sie sich vorstellen eine zweihändige Interaktion in Ihrem Arbeitsalltag am PC anzuwenden?) des Fragebogens. Die Ergebnisse dieser Fragen in der Auswertung lassen den Schluss zu, dass die grundlegende Akzeptanz mit zwei Mäusen zu arbeiten gegeben ist, die Alltagstauglichkeit jedoch teils kritisch gesehen wird. Viele der Probanden zeigten sich fasziniert von der Idee, eine Anwendung oder ein Spiel mit zwei Mäusen zu steuern. Nichtsdestotrotz scheint die Gewöhnung an den Status quo so groß zu sein, dass alternative Lösungen eher schwer durchsetzbar sind. Dies hängt vermutlich sowohl mit technischen Problemen unter dem am häufigsten genutzten Betriebssystem, Microsoft Windows, als auch mit den Gewohnheiten der Menschen zusammen, die täglich mit PCs arbeiten.

### *Präzision*

Nach Auswertung der Messdaten kann man behaupten, dass beide Interaktionsmodelle in ausreichendem Maße präzise sind. Zwar hat keiner der Probanden eine pixelgenaue Übereinstimmung der Objekte mit ihren Referenzobjekten erreicht, dies war jedoch auch nie das Ziel des Testfalls. Denn auch in verbreiteten Bildbearbeitungs- und Layouting-Programmen, wie beispielsweise Adobe Photoshop, Adobe Illustrator oder Adobe InDesign, die man defacto als derzeitigen Industriestandard bezeichnen kann, wird selten ein Objekt ohne Hilfsmittel mit der Maus pixel- oder punktgenau platziert. Oft verwendet man Hilfslinien (Snap to Guide), Dokumentraster (Snap to Grid), andere intelligente Einrastfunktionen oder Tastatureingaben, um die letztendliche Transformation zu realisieren. Solcherlei Hilfsmittel wurden in der Testanwendung bewusst nicht angeboten, da ansonsten keine Präzisionsbestimmung hätte stattfinden können. Stellt man sich jedoch vor, dass die Anwendung mit einem oder mehreren solcher Systeme ausgestattet worden wäre, so kann man mit Gewissheit sagen, dass die Resultate in nahezu allen Fällen hundertprozentig präzise gewesen wären. Dieses Kriterium wird also von beiden Interaktionsmodellen mindestens genauso gut erfüllt, wie von herkömmlichen Ansätzen. Abschließend bleibt zu bemerken, dass der asymmetrische Ansatz bei der Präzision aufgrund seines getrennten Bedienablaufs (nur eine Transformationsart zur Zeit) etwas besser abgeschnitten hat als der symmetrische.

### *Effizienz*

Was den Vergleich der Effizienz beider Modelle angeht, unterscheiden sich die Resultate aus der quantitativen und der qualitativen Evaluation. Folgt man der Auswertung der Fragebögen, so hat der überwiegende Anteil an Testnutzern den Eindruck gehabt, mit dem asymmetrischen Modell effizienter zu arbeiten.

Dies wird jedoch durch die Fakten der Messdaten widerlegt. Die benötigte Zeit zur Absolvierung eines Aufgabendurchlaufs sank im symmetrischen Modell mit fortschreitender Eingewöhnung. Beim asymmetrischen Modell war dies nur bedingt der Fall. Außer im ersten Durchlauf benötigten die Probanden im asymmetrischen Modell immer mehr Zeit, um die Aufgabe zu erfüllen. Die Anzahl von Mausklicks ist in diesem Modell ebenfalls durchschnittlich etwas größer. Das einzige Kriterium, das mit in die Effizienz einfließt und für das asymmetrische Modell spricht, ist die erreichte Präzision. Jedoch sind die Unterschiede hier durchschnittlich dermaßen gering ausgefallen, dass auch dies vernachlässigbar erscheint. Im Gesamtergebnis ist die Beziehung zwischen Kosten und Nutzen im symmetrischen Modell vorteilhafter – und damit auch seine Effizienz größer.

### *Intuitivität*

Die Intuitivität beider Modelle ließ sich in erster Linie durch die qualitative Evaluation ermitteln. Insbesondere spielte hier Frage 8 (Welches Bedienungsmodell war für Sie intuitiver zu handhaben?) eine Rolle. Die überwiegende Anzahl an Probanden beurteilte diese Frage zu Gunsten des symmetrischen Modells, wobei die Gewichtung widererwartend nicht sehr eindeutig ausfiel. Hier hatte ich eine deutlich größere Bevorzugung des symmetrischen Modells prognostiziert. Obwohl dieses Modell eine direkte Vergleichbarkeit mit der Metapher des alltäglichen Anfassens und Bewegens von Objekten nahelegt, scheint die Bekanntheit von einer Art Werkzeugpalette so weit fortgeschritten zu sein, dass viele auch das asymmetrische Modell mit der Toolglass-Technik als intuitiv genug erachteten. Nichtsdestotrotz zeigte sich, dass die Intuitivität des symmetrischen Modells generell besser war. Dies begründet sich insbesondere durch die größeren Eingewöhnungsprobleme einiger Probanden beim asymmetrischen Modell. Auch die Messdaten zeigen, dass hier des Öfteren versucht wurde, mit der nichtdominanten Hand zu klicken, obwohl die Maus zur Steuerung des Toolglass niemals eine Klickfunktion besaß.

### *Integration der nichtdominanten Hand in den Bedienablauf*

Die Integration der nichtdominanten Hand in die Benutzung während der Tests ließ sich gut aus den Messdaten bestimmen und wurde nicht gesondert auf dem Fragebogen abgefragt. Die Ergebnisse der quantitativen Evaluation lassen darauf schließen, dass die nichtdominante Hand im symmetrischen Modell besser integriert wurde als im asymmetrischen. Dafür sprechen insbesondere die Verhältnisse der Mauswege und Mausbewegungshäufigkeiten. Im asymmetrischen Modell wurde die rechte Maus durchschnittlich doppelt so oft genutzt wie im symmetrischen Modell. Auch die Mauswege der rechten gegenüber der linken Maus sind beim asymmetrischen Modell bis zu einem Drittel länger. Zusätzlich könnte man die Verteilung der Klicks, die durch das symmetrische Modell stattfindet als Integrationsfaktor ansehen. Da dies jedoch vordefinierte Eigenschaften beider Modelle waren, möchte ich dieses Kriterium hier außer Acht lassen.

## 4.4 Interpretation im Hinblick auf Guiard

Im Hinblick auf Guiards Arbeit [Guiard, Y. (1987)], legt die Studie auf den ersten Blick nahe, dass auch ein völlig symmetrischer Ansatz für eine zweihändige Bedienung scheinbar problemlos gemeistert wird und – in dem in dieser Studie behandelten, speziellen Szenario – scheinbar sogar besser, also effizienter funktioniert als der asymmetrische Ansatz. Dem ist jedoch bei genauerem Hinsehen nicht mehr so. Guiard zeigt in seiner Arbeit auf, dass beide Hände des Menschen in der Regel als »kinematische Kette« ("kinematic chain") zusammen funktionieren. Das heißt: die nichtdominante Hand folgt in ihrer Arbeit stets der dominanten. Dieses Prinzip eignet sich verständlicherweise besonders gut für bereits trainierte, asymmetrische Aufgabenstellungen. Der dieser Studie zu Grunde liegende symmetrische Ansatz lässt dem Nutzer nun aber generell die Freiheit, dieses Prinzip auch hier anzuwenden. Der Benutzer wird nicht gezwungen, für eine Aufgabe wirklich mit beiden Händen das gleiche zu tun, sondern beide Hände haben lediglich die gleichen Möglichkeiten der Manipulation. Beispielsweise kann der Benutzer im symmetrischen Modell dieser Studie durchaus die nichtdominante Hand nur zum Auswählen eines Rotations- oder Skalierungsreferenzpunkts verwenden. Tatsächlich zeigte es sich durch die Messergebnisse der quantitativen Evaluation und meiner Beobachtungen der Probanden während der Durchführung der Aufgabe, dass viele anfangs darauf verzichteten, beide Hände gleichmäßig einzusetzen. Diese Handlungsweise veränderte sich zwar im Laufe der Aufgabendurchläufe dahingehend, dass die Probanden die nichtdominante Hand häufiger und weiter bewegten, jedoch blieb es gerade bei der Ausübung der eigentlichen Transformation oft bei der hauptsächlichen Nutzung der rechten Hand, um die wirkliche Arbeit zu erledigen. Guiards Theorie scheint also auch in diesem Modell anwendbar zu sein.

## 5 Fazit

Die Studie hat noch einmal demonstriert, dass zweihändige Interaktionskonzepte funktionieren und damit die Ergebnisse der von Myers und Buxton [Buxton, W. (1986)] durchgeführte Studie bestätigt. Des Weiteren wurde gezeigt, dass ein symmetrischer Ansatz im Bereich der zweihändigen Bedienung durchaus besser funktionieren kann als ein asymmetrischer. Dies wird jedoch durch die Eigenschaft des hier getesteten symmetrischen Ansatzes eingeschränkt, dass die Möglichkeit für den Benutzer besteht, sich vor der realen symmetrischen Nutzung beider Hände – möglicherweise unterbewusst – zu drücken (siehe 4.4). Des Weiteren ist ein symmetrischer Ansatz sicher nicht in jedem Anwendungsszenario möglich oder sinnvoll. Folgt man Guiard [Guiard, Y. (1987)], so sind vermutlich wirklich die meisten alltäglichen Aufgaben, die Menschen mit beiden Händen erledigen von einer tief verwurzelten Asymmetrie geprägt. Nichtsdestotrotz stellt diese Studie fest, dass zumindest ein symmetrisches, beidhändiges Interaktionsmodell durchaus mit dem populäreren asymmetrischen Ansatz, der Toolglass-Technik, mithalten kann.

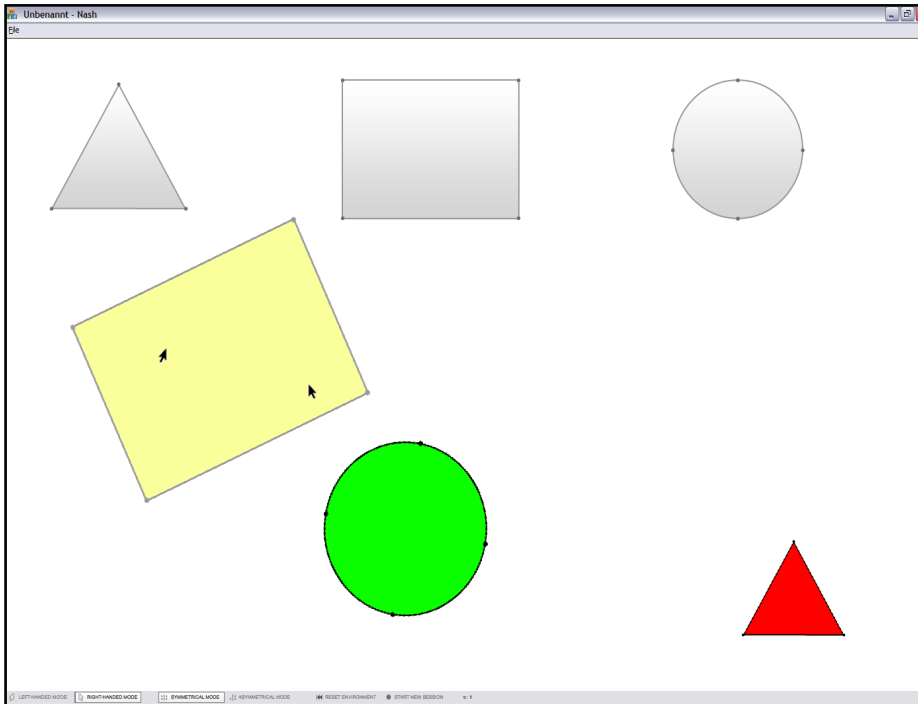
## 6 Literaturverzeichnis

1. Beaudouin-Lafon, M.; Lassen, H. M. (2000): The Architecture and Implementation of CPN2000, A Post-WIMP Graphical Application. In: Proceedings of the 13th annual ACM symposium on User interface software and technology, ACM Press, New York, NY, USA, pp. 181-190.
2. Bier, E. A.; Stone, M. S.; Pier, K.; Buxton, W.; DeRose, T. D.; (1993): Toolglass and Magic Lenses: The See-Through Interface. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, ACM Press, New York, NY, USA, pp. 73-80.
3. Buxton, W.; Myers, B. (1986): A study in two-handed input, Proceedings of the SIGCHI conference on Human factors in computing systems, Boston, Massachusetts, USA, pp. 321-326.
4. Guiard, Y. (1987): Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model. *Journal of Motor Behavior*, 1987, 19, pp. 486-517.
5. Kabbash, P.; Buxton, W.; Sellen, A. (1994): Two-handed Input in a Compound Task. In: Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence, ACM Press, New York, NY, USA, pp. 417-423.
6. Microsoft Corporation, DirectInput (2005): IDirectInput8 Interface. (DirectInput Reference from the DirectX 9.0c SDK.), [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9\\_c/IDirectInput8\\_EnumDevices.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/IDirectInput8_EnumDevices.asp), Abruf am 14.08.2006.
7. Microsoft Corporation, MFC (2006): MSDN Library, Microsoft Foundation Class Library, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vcmfc98/html/mfchm.asp>, Abruf am 12.08.2006.
8. Schneiderman, B. (1983): Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8), pp. 57-69.
9. Smith, D. C.; Irby, C.; Kimball, R.; Verplank, W.; Harslem, E. (1982): Designing the Star user interface. *Byte*, 7(4), pp. 242-282.
10. SourceForge.net (2006): SourceForge.net. <http://www.sourceforge.net>, Abruf am 12.08.2006.
11. Westergaard, M. (2002): Supporting Multiple Pointing Devices in Microsoft Windows. In: Proceedings of 24th Microsoft Summer Workshop for Faculty and PhDs. Cambridge, England, September 2002. (Online verfügbar: <http://klafbang.eu/personlig/publications/mouse.pdf>, Abruf am 14.08.2006.)
12. Wotsit.org (2006): The Programmer's File Format Collection. Suchwort: "CSV", <http://www.wotsit.org>, Abruf am 20.08.2006.

# APPENDIX I: Der Fragebogen

Fragebogen		Second Hand Interface Eine Vergleichsstudie zweihändiger Interaktionsmodelle			
1.	Wie oft arbeiten Sie im Alltag mit dem PC?	sehr oft	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	sehr selten	<input type="checkbox"/> k.A.
2.	Wie viel Erfahrung haben Sie mit herkömmlichen Bildbearbeitungs oder Layouting-Programmen?	sehr viel	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	keine	<input type="checkbox"/> k.A.
3.	Haben Sie praktische Erfahrung mit ähnlichen zweihändigen Bedienungsformen?	viel	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	keine	<input type="checkbox"/> k.A.
4.	Wie lange haben Sie gebraucht, um das Konzept der zweihändigen Bedienung zu verstehen?	kurz	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	lange	<input type="checkbox"/> k.A.
5.	War es für Sie schwierig sich an die Bedienung mittels zwei Mäusen zu gewöhnen?	sehr	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	gar nicht	<input type="checkbox"/> k.A.
6.	Fanden Sie die Bedienung mittels zwei Mäusen angenehm?	sehr	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	gar nicht	<input type="checkbox"/> k.A.
7.	Welches Bedienungsmodell sagte Ihnen mehr zu? (A = erstes; B = zweites Modell)	A	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	B	<input type="checkbox"/> gleich <input type="checkbox"/> k.A.
8.	Welches Bedienungsmodell war für Sie intuitiver zu handhaben?	A	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	B	<input type="checkbox"/> gleich <input type="checkbox"/> k.A.
9.	Hatten Sie generell das Gefühl mit der zweihändigen Bedienung effizient zu arbeiten?	sehr	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	gar nicht	<input type="checkbox"/> k.A.
10.	Bei welchem Bedienmodell hatten Sie das Gefühl effizienter zu arbeiten?	A	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	B	<input type="checkbox"/> gleich <input type="checkbox"/> k.A.
11.	Könnten Sie sich vorstellen eine zweihändige Interaktion in Ihrem Arbeitsalltag am PC anzuwenden?	ja	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	nein	<input type="checkbox"/> k.A.
12.	Was hat Ihnen besonders gefallen / nicht gefallen? Haben Sie Kommentare?	<hr/> <hr/> <hr/> <hr/> <hr/>			
Alter: _____		Fortlaufende Identifikationsnummer: _____			
Geschlecht: <input type="checkbox"/> männlich <input type="checkbox"/> weiblich		<input type="checkbox"/> rechtshänder <input type="checkbox"/> linkshänder			
<hr/> <b>Second Hand Interface • Bachelor Report • SoSe 2006 • Universität Bremen • Tobias Lensing</b>					

## APPENDIX II: Screenshot der Testanwendung



## APPENDIX III: Hinweise zu der beiliegenden CD

Auf der beiliegenden CD befinden sich die Auswertungsdateien sowie der gesamte Quellcode der für die Studie entwickelten Testanwendung inklusive der ihr unterliegenden Bibliotheken.

Um die Testanwendung (NASH.EXE) zu starten, werden folgende Werkzeuge benötigt:

- Visual C++ 2005
- DirectX 9.0c
- PC mit DirectX 9 kompatibler Grafikkarte und min. 256 MB RAM

Damit die Testanwendung funktioniert, müssen zwei Mäuse, die mit dem CPN-Maustreiber installiert sind, an das System angeschlossen sein. Weitere Hinweise siehe readme.txt auf der CD.

Zur Kompilierung der Testanwendung wird darüber Hinaus das DirectX 9 SDK sowie ein aktuelles Platform SDK von Microsoft benötigt.

Die kompilierte Testanwendung findet sich auf der CD unter bin\debug\nash.exe.

Die Solution-Datei für die Quellcodeprojekte heißt: solutions\nash\nash.sln

Die endgültigen Quelldateien der Auswertung befinden sich im Verzeichnis: bin\debug\evaluation