Kapitel 5

Integration multidimensionaler Interaktionsgeräte

5.1 Einleitung

Um mit virtuellen Umgebungen zu interagieren, werden Interaktionsgeräte benötigt, die dem Benutzer die Interaktion in dreidimensionalen Szenen erlauben. In den letzten Jahren wurde eine ganze Palette solcher Geräte entwickelt. Diese erlauben die direkte Eingabe von 3D-Positionen und Orientierungen, intuitive Interaktion etwa durch Gesten- oder Spracherkennung und beziehen zusätzlich zum Sehsinn auch das Gehör und den Tastsinn mit ein. Grafische Displays erzeugen räumlich wirkende Bilder durch die stereoskopische Darstellung, die für beide Augen getrennt perspektivisch korrekte Bilder darstellt. Während die Ansteuerung grafischer Displays durch den grafischen Renderer geleistet wird, wird für die Anbindung von Interaktionsgeräten ein Softwaresystem benötigt, welches die hardwareunabhängige Handhabung dieser Geräte ermöglicht. Ein solches Softwaresystem wurde im Rahmen dieser Arbeit entwickelt und soll in diesem Kapitel vorgestellt werden [FrRo00].

Zuvor werden aber in den folgenden Abschnitten die Interaktionsgeräte vorgestellt und klassifiziert. Anschließend werden technische Aspekte und Probleme bei deren Handhabung diskutiert und Forderungen an ein System zur Integration abgeleitet. Daraufhin werden bestehende Systeme untersucht und bewertet. Darauf aufbauend wird das IDEAL System vorgestellt und diskutiert. Das Kapitel schließt mit einer Zusammenfassung.

5.2 Interaktionsgeräte

Interaktionsgeräte ermöglichen dem Benutzer mit einem Computersystem, oder genauer mit der darauf laufenden Software zu interagieren. Grundsätzlich kann man zwischen Eingabegeräten und Ausgabegeräten unterscheiden. Eingabegeräte ermöglichen dem Benutzer, Kommandos und Daten zu erzeugen und mit ihrer Hilfe ein Softwaresystem zu steuern. Ausgabegeräte hingegen stellen Informationen über den aktuellen Systemzustand dar. Beispiele für traditionelle Eingabegeräte sind Maus und Tastatur, das klassische Ausgabegerät ist der Bildschirm. Mit dem Aufkommen der interaktiven dreidimensionalen Computergrafik wurden eine Reihe neuartiger Interaktionsgeräte entwickelt. Ihnen ist gemeinsam, daß sie wesentlich mehr Freiheitsgrade besitzen und damit die direkte Interaktion im dreidimensionalen Raum ermöglichen.

Besondere Bedeutung haben dabei Geräte, die Position und Orientierung im dreidimensionalen Raum darstellen, insgesamt also über 6 Freiheitsgrade verfügen. Diese Geräte werden in den folgenden Ausführungen als 6D-Geräte bezeichnet.

Einige Geräte stellen Mischformen dar, sie ermöglichen Eingaben und versorgen den Benutzer gleichzeitig mit grafischer oder haptischer Rückkopplung. Als Beispiel kann die Monitorbrille genannt werden.

5.2.1 Eingabegeräte

Immobile Geräte für den Einsatz am Schreibtisch

Tastatur und Maus Tastatur und Maus sind die klassischen Eingabegeräte für die Interaktion mit textbasierten oder zweidimensionalen grafischen Benutzungsoberflächen. Zwar sind sie für die Interaktion mit dreidimensionalen Darstellungen nur bedingt geeignet, wegen Ihrer allgemeinen Verfügbarkeit spielen sie aber nach wie vor eine wichtige Rolle und werden von allen Interfaces unterstützt. In Kombination mit den Maustasten kann der zweidimensionale Aktionsraum der Maus oder der Cursortasten in den dreidimensionalen Raum abgebildet werden.

Spacemouse und Spaceball Ähnlich wie die Maus finden Spacemouse und Spaceball auf dem Schreibtisch neben der Tastatur Platz. Beide Geräte besitzen einen Griff, der mit der Hand verschoben und gedreht werden kann.

Dieser ist bei der Spacemouse als flacher Puck gestaltet, während er beim Spaceball kugelförmig ist. Allen immobilen Geräten ist gemeinsam, daß sie bauartbedingt *relative* Daten liefern, also keine direkten Positionen und Ori-

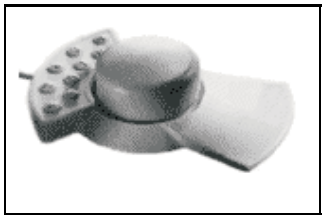

Abbildung 5.1: DLR Spacemouse

entierungen, sondern Positionsänderungen. Etwa leitet das vorwärts drücken des Pucks der Spacemouse eine Vorwärts*bewegung* ein. Mit Hilfe der Spacemouse kann der Benutzer durch das verschieben und verdrehen des Pucks gleichzeitig Translationen und Rotationen steuern, es handelt sich also um ein 6D-Gerät. Sie eignet sich damit ausgezeichnet für die Interaktion in dreidimensionalen computergenerierten Räumen.

Am Körper montierte mobile Geräte

Geräte die am Körper getragen oder an der Kleidung montiert sind, werden hauptsächlich für die immersive Interaktion verwendet. Voraussetzung für die immersive Interaktion ist ein geeignetes Ausgabegerät. Die am Körper montierten oder getragenen Geräte messen etwa Positionen und Orientierungen von Kopf und Händen, sowie die Beugung der Fingergelenke. Hier handelt es sich also um *absolute* Gerätedaten im Unterschied zu den im vorherigen Abschnitt beschriebenen Eingabegeräten.

Datenhandschuh Der Datenhandschuh (eng. Dataglove) wird wie ein normaler Handschuh getragen. Bis zu 22 Sensoren messen die Stellung der Finger. Aus diesen Meßwerten kann ein dreidimensionales Modell der Hand errechnet werden. Durch Auswertung der Fingerstellungen können Gesten erkannt und Handhabungsoperationen wie z.B. Greifen und Zeigen nachgebildet werden.

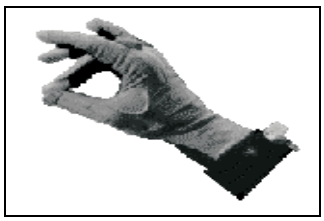

Abbildung 5.2: Virtual Technologies Cyberglove

Trackingsysteme Zur Registrierung von Körperbewegungen werden Trackingsysteme eingesetzt. Diese bestehen aus einer Reihe von Sensoren (oft verkürzt 'Tracker' genannt), die am Körper oder an der Ausrüstung, etwa dem Datenhandschuh oder der Monitorbrille, angebracht sind.

Man unterscheidet verschiedene Meßverfahren:

- elektromagnetische
- optische
- andere Verfahren wie Ultraschall und Trägheitsmoment

Elektromagnetische Trackingsysteme bestehen aus einer elektrischen Magnetquelle und einer Reihe von Sensoren, die das erzeugte Magnetfeld messen und die Meßwerte über Kabel an die Zentraleinheit des Systemes zurück liefern. Diese berechnet dann Position und Orientierung jedes Sensors relativ zur Magnetquelle. Bei Trackingsystemen handelt es sich typischerweise um 6D-Geräte, einige Systeme liefern aber nur Positionen oder nur Orientierungen. Die Kabelverbindung der Sensoren mit der Zentraleinheit stellt einen Nachteil dieser Systeme dar. Sie begrenzt die Bewegungsfreiheit der Benutzer. Dies ist besonders problematisch, wenn der Benutzer eine Monitorbrille trägt, seine reale Umwelt also nicht wahrnimmt und so leicht über die Kabel stolpern kann.

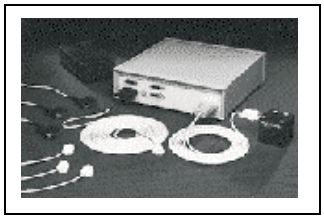

Abbildung 5.3: Polhemus Fastrak

Optische Trackingsysteme kommen ohne Kabel aus. Sogenannte Marker ersetzen die Tracker der elektromagnetischen Systeme. Marker sind typischerweise kleine Kugeln oder Scheiben, die Licht in einem bestimmten Spektralbereich stark reflektieren und werden, wie die Tracker am Körper des Benutzers angebracht. Ein System aus ein oder mehren Kameras und einer Zentraleinheit erkennt zunächst die Markerpositionen im Bild (die sogenannte Segmentierung) und errechnet dann aus Kombinationen von Bildern verschiedener Kameras die angenommenen dreidimensionalen Positionen der Marker im Raum. Um Orientierungen zu erkennen, müssen entweder drei oder mehr Marker miteinander fest verbunden werden, oder ein Modell der getrackten

Person zu Grunde gelegt werden. Problematisch bei optischen Trackingsystemen sind Verdeckungen. Wird ein Marker von zu vielen Kameras nicht mehr gesehen, kann keine korrekte Position berechnet werden.

Werkzeuge Eingabegeräte für immersive Anwendungen können nicht nur am Körper, sondern auch in der Hand getragen werden. Sie erfüllen dann die Funktion eines Interaktionswerkzeuges. Solche Werkzeuge werden in den meisten Fällen zum Zeigen benutzt. Objekte in einer virtuellen Umgebung werden identifiziert oder markiert. Abhängig vom Interaktionsparadigma müssen Algorithmen für Strahltests und Kollisionserkennung bereitgestellt werden. Ein kommerzielles Gerät stellt der Polhemus Stylus dar. Neben einem integrierten Tracker besitzt er eine Taste, mit der zusätzlich zur Positionsbestimmung Aktionen, wie etwa eine Markierung ausgelöst werden können.

Im Rahmen eines Projektes, in dem die unterirdischen Kultstätten von Dunhuang in China als virtuelles Modell rekonstruiert wurden ([LuWe99]), wurde am Fraunhofer-IGD wurde eine virtuelle Taschenlampe entwickelt. In das Gehäuse einer Taschenlampe wurde ein Tracker eingebaut, zusätzlich kann die Position des Schalters kann durch eine Steuerelektronik abgefragt werden. Diese ist wiederum über eine serielle Schnittstelle mit einem Rechner verbunden. In der virtuellen Umgebung kann eine simulierte Lichtquelle mit Hilfe der Taschenlampe bewegt werden. Mit dieser Taschenlampe können die Höhlen von Dunhuang nun von Besuchern (virtuell) erforscht werden.

Spracheingabe Mit dem Aufkommen von leistungsfähigen Spracherkennungsystemen ist eine neue und leistungsfähige Mensch-Maschine Schnittstelle entstanden, die auch in der Virtuellen Realität angewendet wird. Gesprochene Befehle erlauben die direkte Manipulation der virtuellen Umgebung.

5.2.2 Ausgabegeräte

Haptische Ausgabegeräte

Nach [Zieg98] erfüllen haptische Ausgabegeräte (auch haptische Displays genannt) zwei Aufgaben:

1. Darstellung bzw. Vermittlung von haptischen Reizen und Positionsänderungen

2. Messung von Position und Kontaktkräften (sowie deren zeitlicher Änderung) der Hand oder anderer Körperteile.

Im Gegensatz zu den anderen in dieser Arbeit betrachteten Interaktionsgeräten kommunizieren haptische Ausgabegeräte bidirektional mit der Anwendung. Die vom haptischen Display ermittelte Position ist an die Anwendung zu übertragen, da die von dieser zu simulierende Kontaktkraft von der Position der Hand bzw. eines Instrumentes abhängig ist. Das gleiche gilt für die vom Gerät gemessene Kontaktkraft. Für die Anwendung wiederum kann es erforderlich sein, die Hand oder das Instrument an eine bestimmte Position zu bringen und die vom Gerät darzustellende Kraft zu steuern. Dies erfordert eine Datenübertragung von der Anwendung an das haptische Ausgabegerät.

Bei der Kommunikation zwischen Anwendung und haptischem Display sind besondere Anforderungen an die Geschwindigkeit der Datenübertragung zu erfüllen. Während bei Eingabegeräten eine Übertragungsrate entsprechend der Rate der Bildgenrierung von 50 Hz ausreicht, ist für haptische Displays eine Frequenz von bis zu 10000 Hz erforderlich [Zieg98], die mit herkömmlichen seriellen Schnittstellen wie RS232 nicht zu erreichen ist [1]. Ein weiteres Problem entsteht bei der Anwendung selbst: Diese muß in der Lage sein, die physikalische Simulation mit der selben Frequenz durchzuführen. Typischerweise ist die Geschwindigkeit, mit der die Simulationsschleife innerhalb einer Echtzeit-Grafikanwendung läuft, aber durch die Geschwindigkeit der grafischen Darstellung determiniert, die in der Größenordnung von 50 Hz liegt. Daher ist es sinnvoll, die eigentliche Anwendung von der Ansteuerung des haptischen Gerätes zu trennen und beide auf zwei parallel laufende Prozesse zu verteilen. Dies leisten verteilte Architekturen zur Handhabung von Interaktionsgeräten wie INTO (siehe Abschnitt 2.2.2) und IDEAL).

Kraftrückkopplung Kraftrückkopplungsgeräte vermitteln die erzeugte Kontakt- oder Reaktionskraft, indem Körperteile (Finger, Hand, Arm) an eine andere Position gebracht werden [Zieg98]. Der Benutzer muß Gegenkraft aufwenden, um die gewünschte Position aufrecht zu halten. Meist werden kleine leistungsfähige Elektromotoren verwendet. Besitzen diese Motoren eine sehr hohe Dynamik, können auch Vibrationen erzeugt werden. Beispielhaft soll hier das Phantom erwähnt werden:

Das PHANToM (Personal HAptic iNTerface Mechanism) wurde am Massachusetts Institute of Technology (MIT) entwickelt [MaSa94]. Der Anwender nimmt die Kräfte entweder an der Fingerspitze, die er in einen Fingerhut steckt wahr, oder durch einen Stift, der in den Fingerhut gesteckt wird . Das Gerät besitzt drei Freiheitsgrade, die über drei Elektromotoren realisiert wer-

[1] Für das Phantom müssen beispielsweise bis zu 1.4 Mbit/sec erreicht werden. Die Rate ergibt sich wie folgt: Die Größe des zu übertragenen Datensatzes ist 9 mal 16-bit Integer-Werte (vergl. [Zieg98]), 10000 Sätze pro Sekunde.

den. Die Maximalkraft ist mit 10 N, die kontinuierlich Rückkopplungskraft mit 1.5 N angegeben. Die verbleibende Reibungskraft im Leerlauf ist 0.1 N [Zieg98].

Taktile Rückkopplung Wahrnehmungen wie das Ertasten von Oberflächentexturen können mit sogenannten taktilen Displays erzeugt werden. Taktile Displays reizen die Rezeptoren, die für den Tastsinn verantwortlich sind über direkten Hautkontakt.

5.3 Aspekte der Geräteintegration und abgeleitete Forderungen

Ein generelles Problem stellt die Ansteuerung der physikalischen Geräte dar. Da Interaktionsgeräte in Bezug auf Anwendungszweck und Datenformate äußerst unterschiedlich sind, benutzt jeder Hersteller ein eigenes Protokoll. Dies ist selbst bei gleichartigen Geräten unterschiedlicher Hersteller der Fall. Um die Integration hardwareunabhängig zu gestalten, muß ein System also ein einheitliches API (Application Programmers Interface) zur Verfügung stellen, daß von den gerätespezifischen Eigenheiten abstrahiert.
Bei der Standardisierung von grafischen Systemen (GKS [Iso86], und PHIGS [Iso89]) wurde der Begriff der logischen Eingabeklassen geprägt, um Anwendungen von gerätespezifischen Eigenschaften verschiedener Eingabegeräte zu entkoppeln (siehe auch ([Felg95] Seite 101ff). Bei diesen Systemen fordert die Anwendung ein logisches Gerät einer bestimmten Eingabeklasse an, und greift anschließend darüber auf das tatsächlich physikalisch vorhandene Gerät zu. Auch wenn sich das Interaktionsmodell ebenso wie Art und Zahl der Interaktionsgeräte seitdem verändert hat, bleibt das Konzept des logischen Gerätes weiterhin gültig und wird auch in den meisten heute verfügbaren Systemen eingesetzt.

Forderung 1 Ein System zur Integration von Interaktionsgeräten muß eine hardwareunabhängige logische Geräteschnittstelle besitzen.

In vielen Anwendungsfällen kommen verschiedene Geräte für den Einsatz in Frage. Etwa kann für die Navigation eine normale Maus, als auch eine Spacemouse eingesetzt werden. Welches Gerät tatsächlich benutzt wird, wird meist nicht bei der Programmierung einer Anwendung entschieden, sondern erst bei der Installation oder kurz vor dem Start. Daher muß es muß es möglich sein, jede Anwendung extern zu konfigurieren. In vielen Anwen-

dungsfällen kommen verschiedene Geräte für den Einsatz in Frage. Etwa kann für die Navigation eine normale Maus, als auch eine Spacemouse eingesetzt werden. Welches Gerät tatsächlich benutzt wird, wird meist nicht bei der Programmierung einer Anwendung entschieden, sondern erst bei der Installation oder kurz vor dem Start. Daher muß es muß es möglich sein, jede Anwendung extern zu konfigurieren.

Forderung 2 Die Abbildung von logischen auf physikalische Geräte beim Start des Systems über externe Konfigurationsdateien erfolgen.

Tastatur, Maus und Bildschirm sind allgemein verfügbar gehören zur Standardausstattung jedes modernen Computers. Vorteilhaft ist, daß Ihr Gebrauch praktisch jedem Benutzer geläufig ist und sie überall zu finden sind. Für die Interaktion in Virtuellen Umgebungen sind sie jedoch nur bedingt geeignet. Problematisch ist ihre geringe Zahl von Freiheitsgraden. Mit Hilfe der Maus können nur Punkte in einer zweidimensionalen Fläche angefahren werden. Die Maustasten und die Tastatur erlauben eine 1-aus-N Auswahl. Um dennoch mit Hilfe von Maus und Tastatur in dreidimensionalen Szenen zu interagieren, werden Metaphern wie die Virtuelle Kugel ([Felg95] Seite 85) verwendet. Solche Metaphern müssen von der Integrationssoftware bereitgestellt werden, damit die Abbildung logischer Geräte auf Tastatur und Maus möglich ist.

Forderung 3 Es müssen geeignete Metaphern für die 3D-Interaktion mit Maus und Tastatur zur Verfügung stehen.

Während grafische Displays meist über spezifische Hardware angesteuert werden, die wegen ihrer hohen Anforderungen an Bandbreite direkt in den Computer integriert sind, werden die meisten anderen Interaktionsgeräte über serielle Schnittstellen mit dem Computer verbunden. Für komplexe Systeme werden oft eine ganze Reihe solcher Geräte gleichzeitig benutzt. Hier kann es zu Problemen kommen, wenn der Computer nicht über genügend Schnittstellen zum Anschluß solcher Geräte verfügt. Dieses Problem kann gelöst werden, indem über ein lokales Kommunikationsnetzwerk (LAN) auf Geräte zugegriffen werden kann, die an andere Rechner angeschlossen sind.

Forderung 4 Der netzwerkweite Zugriff auf Interaktionsgeräte muß möglich sein.

Die Benutzung vieler verschiedener Interaktionsgeräte ist ein fehlerträchtiger Vorgang. Geräte können ausgeschaltet oder falsch angeschlossen sein. Jedes Gerät muß einzeln konfiguriert und kalibriert werden. Schließlich kann ein Gerät während des Betriebes ausfallen. Dann ist es auf keinen Fall ein akzeptables Vorgehen, die Anwendung zu beenden, den Fehler zu beheben und danach neu zu starten. Es muß möglich sein, den Fehler zu beheben oder notfalls das Gerät auszutauschen, und dann mit der Anwendung fortzufahren. Um möglichst viele Fehler im Vorfeld zu erkennen und auszuschalten, sollten Test- und Kalibrierungswerkzeuge mit einheitlicher Benutzungsschnittstelle zur Verfügung gestellt werden.

Forderung 5 Im Fehlerfall muß eine Rekonfigurierung oder ein Austausch von Geräten möglich sein, ohne die Applikation zu beenden.

Komplexe Applikationen verwenden in vielen Fällen mehrere gleichartige Interaktionsgeräte, beispielsweise einen Datenhandschuh für die rechte und die linke Hand, um beidhändiges Greifen zu simulieren. Ein System zur Integration von Interaktionsgeräten muß dieses erlauben.

Forderung 6 Die gleichzeitige Verwendung mehrerer gleichartiger Interaktionsgeräte muß möglich sein.

In großen Forschungslaboren ebenso wie in industriellen Institutionen werden viele Arbeiten parallel durchgeführt. In der praktischen Arbeit kommt es oft vor, daß verschiedene VR-Applikationen gleichzeitig laufen und dabei möglicherweise versehentlich auf das selbe physikalische Gerät zugreifen. Dieser Konflikt sollte keinesfalls zur Fehlfunktion der Systeme führen. Auch im Normalbetrieb sollte es möglich sein, neben der eigentlichen VR-Applikation einen Gerätemonitor zur Kontrolle mitlaufen zu lassen (siehe Abschnitt 5.4.9).

Forderung 7 Der parallele Zugriff mehrerer Applikationen auf ein Gerät soll erlaubt sein.

Nachdem nun die Forderungen an ein System zur Integration von Interaktionsgeräten in Anwendungen der Virtuellen Realität aufgestellt worden sind, werden in den folgenden Abschnitten existierende System untersucht und in Bezug die Erfüllung der Forderungen verglichen und bewertet. Anschließend wird das IDEAL System präsentiert, welches alle genannten Forderungen erfüllt.

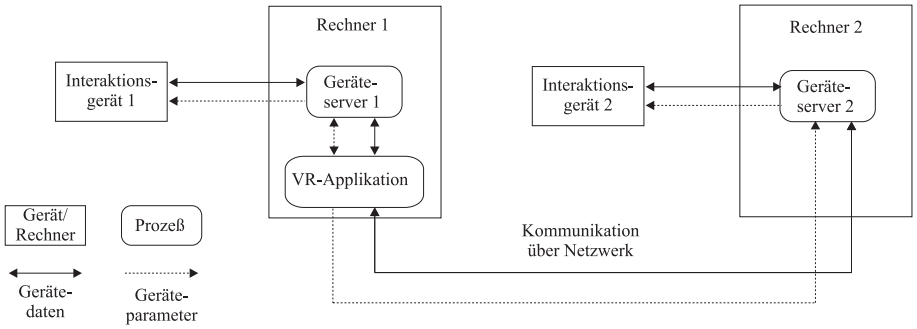

Abbildung 5.4: Datenfluß zwischen Geräteserver und IDEAL -Anwendung

5.4 Das IDEAL System

Basierend auf den vorangegangenen Untersuchungen, Erfahrungen im Umgang mit Interaktionsgeräten und Systemen zu deren Integration wurden Forderungen abgeleitet, die ein solches System idealerweise erfüllen sollte. Da keines der untersuchten Systeme alle Forderungen erfüllen konnte, wurde im Rahmen der vorliegenden Arbeit das IDEAL System entwickelt. Die folgenden Abschnitte beschreiben dessen Systemarchitekur, Geräteabstraktion sowie die Benutzungsschnittstelle. Insbesondere wird darauf eingegangen werden, mit welchen Mitteln alle zuvor aufgestellten Forderungen erfüllt werden.

5.4.1 Systemarchitektur

Um Forderung 4 nach netzwerkweitem Zugriff auf Interaktionsgeräte zu erfüllen, muß der Zugriff auf das physische Interaktionsgerät von der Verarbeitung der Gerätedaten durch die Anwendung getrennt werden. Für jedes Interaktionsgerät übernimmt ein Server die Aufgabe das Gerät zu initialisieren, Daten vom und zum Gerät in dessen spezifischen Protokoll zu übertragen, diese in ein einheitliches Format zu transformieren und an die Anwendung über ein Netzwerk zu übertragen. Ein Server wird auf dem Rechner gestartet an den das Interaktionsgerät angeschlossen ist, während die Applikation auf jedem Rechner im Netzwerk laufen kann. Die Datenübertragung zwischen Server und Applikation erfolgt über Stream-Sockets. Abbildung 5.4 stellt den Datenfluß zwischen Anwendung und zwei Interaktionsgeräten dar. Eines ist lokal und das zweite an einen entfernten Rechner angeschlossen.

Forderung 7 besteht darin, den gleichzeitigen Zugriff mehrerer Applikationen auf ein Interaktionsgerät zuzulassen. Zusätzlich soll ein Gerät nur dann

belegt werden, wenn dessen Daten von einer Anwendung benötigt werden. Dies hat zur Folge, daß es eine Instanz geben muß, welche überwacht welche Server bereits gestartet wurden, wie diese zu erreichen sind und welche Server wieder beendet werden können. Diese Überwachungsfunktion kann auf drei verschiedene Arten realisiert werden.

1. Verwaltung und Überwachung der Server durch die Applikation
   Der Vorteil dieser Variante liegt in ihrer Einfachheit. Jede Applikation startet die von ihr benötigten Geräteserver selbst. Probleme entstehen dann, wenn Anwendung und Server auf verschiedenen Rechnern laufen. Jede Applikation muß auf jedem Rechner, dessen Interaktionsgeräte genutzt werden, Zugriffsrechte zum Starten von Programmen besitzen. Auch kann bei diesem Verfahren nicht ausgeschlossen werden, daß mehrere Applikationen gleichzeitig versuchen, einen Server für dasselbe Gerät zu starten. Damit kann Forderung 7 nicht erfüllt werden.

2. Eine zentrale Instanz innerhalb des Netzwerkes
   Eine zentrale Instanz überwacht und verwaltet alle Geräteserver im Netzwerk. Sie regelt den gleichzeitigen Zugriff mehrerer Applikationen und kann das mehrfache Starten eines Servers ausschließen. Allerdings hat sie den selben Nachteil wie die erste Variante, Zugriffsrechte zum Starten der Server auf allen Rechnern im Netzwerk besitzen zu müssen. Auch ist die unkontrollierte Beendigung eines Servers von der zentralen Instanz aus nicht ohne weiteres festzustellen. Schließlich stellt die zentrale Instanz einen "Single Point of Failure" dar; fällt sie aus, ist die Verwendung von Interaktionsgeräten im gesamten Netzwerk nicht mehr möglich.

3. Eine Überwachungsinstanz auf jedem Rechner mit angeschlossenen Interaktionsgeräten
   Es werden nur die jeweils an einen Rechner angeschlossenen Geräte verwaltet. Das Starten der Server kann auf jedem Rechner lokal erfolgen. Zur Überwachung der Serverprozesse stehen alle Funktionen des Betriebssystems zur Interprozeßkommunikation und Prozeßsteuerung zur Verfügung. Wie bei dem zweiten Verfahren ist der konkurrierende Zugriff mehrerer Applikationen auf ein Interaktionsgerät möglich. Fällt die Überwachungsinstanz auf einem Rechner aus, bleibt der Zugriff auf die an andere Rechner angeschlossenen Geräte weiterhin möglich.

Aufgrund ihrer Vorteile wurde die letzte Variante zur Implementierung des IDEAL -Systems verwendet. Damit besteht IDEAL aus drei wesentlichen Komponenten, die im Folgenden benannt werden:

- Geräteserver
  Prozesse, welche die Schnittstelle zu je einem Interaktionsgerät verwalten und über das Netzwerk mit einer oder mehreren Applikationen Daten austauschen. Für jedes Interaktionsgerät muß ein spezieller Server implementiert werden.

- Demonprozeß (kurz Demon)
  Prozeß, der die Geräteserver auf einem Rechner überwacht und verwaltet. Auf jedem Rechner, an den Interaktionsgeräte angeschlossen sind, muß ein Demon laufen. Anforderungen von Applikation nach Zugriff auf Interaktionsgeräte müssen an diesen Prozeß gerichtet werden.

- Logische Geräteschnittstelle
  Komponente von IDEAL , die als Teil des Anwendungsprogramms läuft. Sie stellt der Applikation eine Klassenhierarchie logischer Geräte zur Verfügung und kommuniziert mit Demon und Geräteservern.

Abbildung 5.5 stellt die Zusammenhänge zwischen Demonprozessen, Geräteservern und logischer Geräteschnittstelle grafisch dar. In dem gezeigten Beispiel laufen zwei VR-Anwendungen auf zwei verschiedenen Rechnern. Jede der Anwendungen kommuniziert mit drei Geräteservern, wobei zwei Server beide Applikationen bedienen. Demonprozesse überwachen und verwalten die Server auf ihren Rechnern. Die Funktion des Geräteverzeichnisses und der Sensordateien wird in Abschnitt 5.4.5 erläutert.

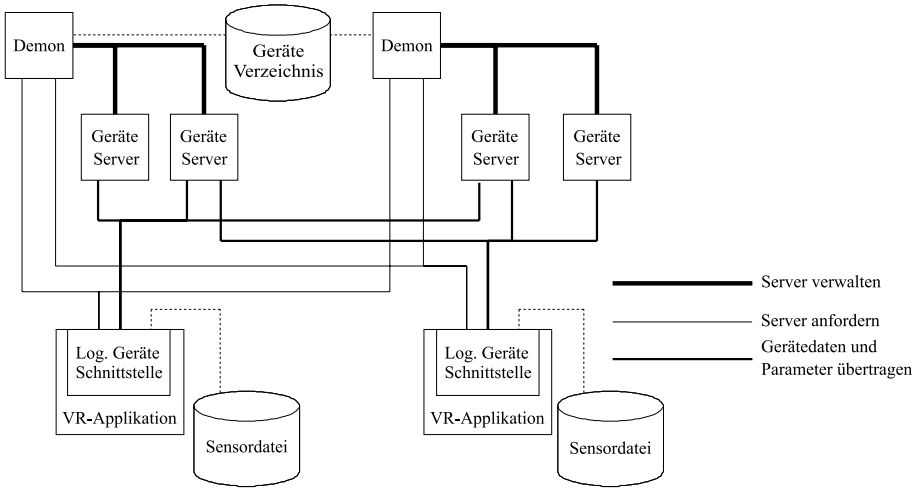

Abbildung 5.5: Datenfluß zwischen Anwendungen, Demonprozessen und Geräteservern

5.4.2 Demon Prozesse

Bei dem Entwurf des IDEAL -Systems stand die Ausfallsicherheit des Systems im Vordergrund. Keinesfalls sollte eine fehlerhafte Konfiguration oder der Ausfall eines Interaktionsgerätes dazu führen, daß die Applikation abgebrochen werden muß. Vielmehr soll es möglich sein, Konfigurationsfehler zu beheben oder das defekte Gerät auszutauschen und mit der Anwendung fortzufahren (Forderung 5). Die Rolle der Überwachungsinstanz wird durch die Demonprozesse übernommen. Der Demon hat darüber hinaus die Aufgabe, Geräteserver zu starten, nicht mehr benötigte Server zu beenden und der logischen Geräteschnittstelle einer Applikation mitzuteilen, unter welcher Socket Port Nummer ein Server zu erreichen ist.

Für die Verwaltung und Überwachung jedes Servers existiert im Demon ein endlicher Zustandsautomat, wie er in Abbildung 5.6 dargestellt ist. Führt die Anforderung einer Anwendung zum Start eines neuen Geräteservers, wird ein neuer Automat erzeugt, nach Beendigung des Servers gelöscht.

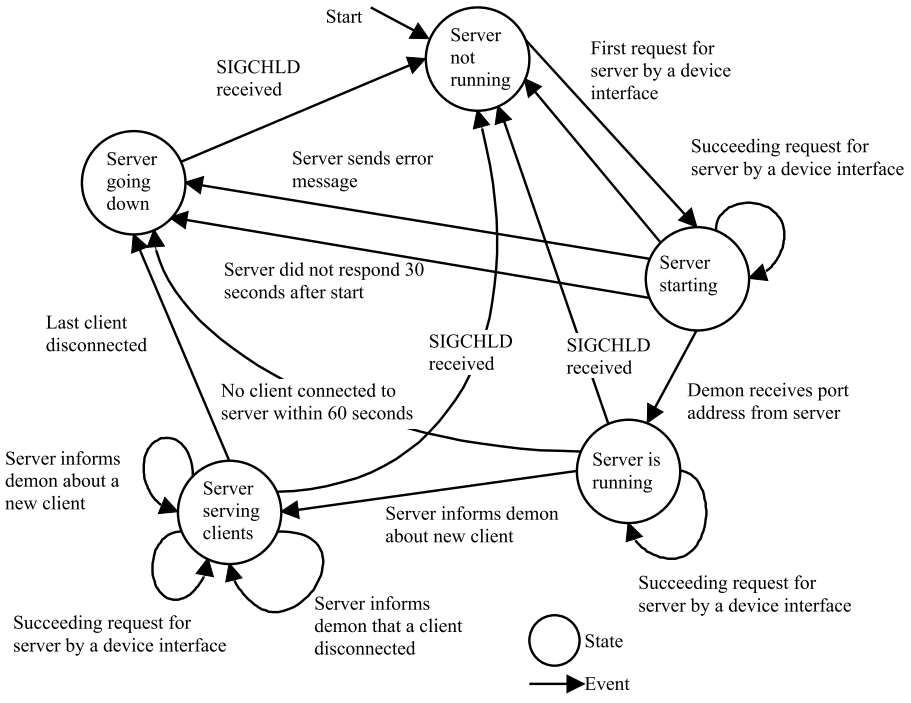

Abbildung 5.6: Zustandsübergangsdiagramm des Demonprozesses

Während der Laufzeit des Servers nimmt der Automat verschiedene Zustände an, die dem aktuellen Status des Servers entsprechen. Zustandsübergänge werden durch die folgenden Ereignisse ausgelöst:

- An-/Abmeldung einer Anwendung,
- Nachrichten vom Server an den Demon,
- SIGCHLD ausgelöst durch Terminierung des Servers und
- Ablaufen von Timern

Beim ersten Anmelden einer Anwendung wird ein neuer Geräteserver gestartet. Melden sich weitere Anwendungen an, reagiert der Server mit dem Erzeugen eines neuen Client-Prozesses, der diese Anwendung mit Daten versorgt. Die umgekehrte Wirkung hat das Abmelden einer Applikation. Beim Abmelden der letzten Applikation kann der Server durch den Demon beendet werden.

Während der Laufzeit des Servers informiert dieser den Demon über Ereignisse, in dem er über eine Socketverbindung Nachrichten an den Demon schickt. Der Automat reagiert darauf mit einer Zustandsänderung. Mögliche Nachrichten sind die Portadresse, über die der Server durch Applikationen erreicht werden kann, das An- und Abmelden einer Anwendung sowie Fehlermeldungen.

Bleiben im Fehlerfall bestimmte Nachrichten aus, führt das Ablaufen von Timern zu einer Reaktion des Demons, der den Server beendet und eine Fehlermeldung an die Applikationen sendet. Diese Fehlermeldung wird auf Seite der Anwendung durch die logische Geräteschnittstelle transparent für die Anwendung verarbeitet. Die Initialisierung des logischen Gerätes wird unterbrochen, und eine Fehlermeldung angezeigt. Der Anwender hat nun die Möglichkeit, den Fehler zu beheben und mit der Anwendung fortzufahren.

5.4.3 Server Prozesse

Um Forderung 7 nach gleichzeitigem Zugriff mehrerer Applikationen auf ein Interaktionsgerät zu erfüllen, wurden die Geräteserver als Multiprozeßserver ausgeführt. Wie Abbildung 5.7 zeigt, wird zunächst der Vaterprozeß (1) des Servers durch den Demon gestartet. Dieser erzeugt sofort den geräteseitigen Prozeß (2). Dieser Prozeß hat die Aufgabe, die Kommunikation mit dem Gerät aufzubauen und dieses zu initialisieren. Auftretende Fehler werden direkt an den Demon gemeldet. Anschließend werden kontinuierlich Gerätedaten gelesen und in einem Shared Memory Segment abgelegt.

Sobald sich eine Applikation beim Server anmeldet (genauer bei dessen Vaterprozeß 1), startet dieser Prozeß 3, mit dem Zweck, diese Anwendung mit Daten zu versorgen. Dieser baut eine Socketverbindung mit der Anwendung auf. Daraufhin werden kontinuierlich Gerätedaten aus dem Shared Memory

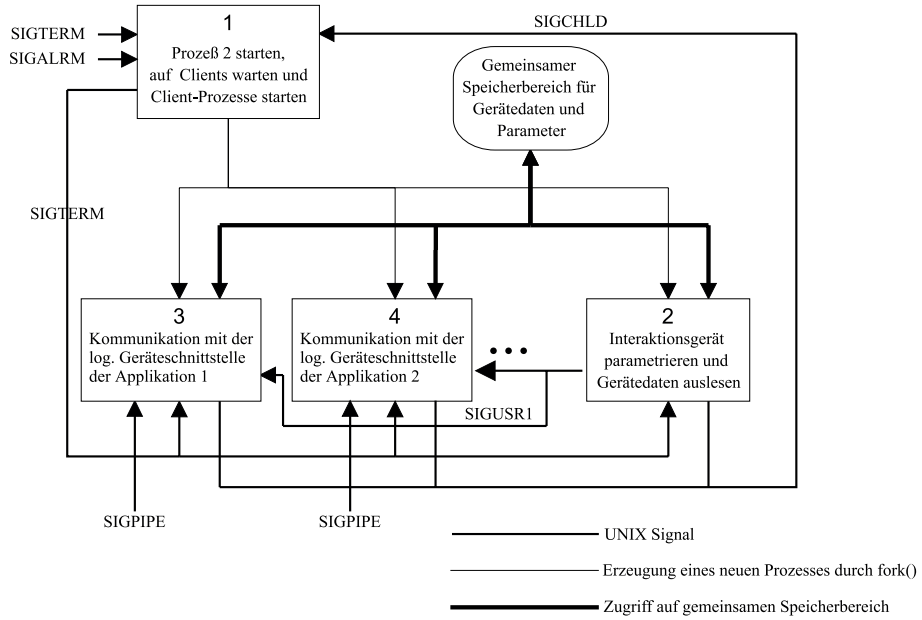

Abbildung 5.7: Ablauf der Prozeßinitialisierung im Server

Segment gelesen, in logische Daten umgewandelt (siehe 5.4.7) und an die Applikation gesendet. Meldet sich eine weitere Applikation an, wird Prozeß 4 gestartet, der diese seinerseits bedient. Die Synchronisation für den Zugriff auf das Shared Memory Segment wird über Locks abgewickelt. Das Vorliegen neuer Gerätedaten teilt Prozeß 2 über das Signal SIGUSR1 mit, was Prozeß 3 und 4 dazu veranlaßt, ihrerseits neue Daten an die Applikationen zu senden.

5.4.4 Die logische Geräteschnittstelle von IDEAL

Aufgabe der logischen Geräteschnittstelle ist es, der Anwendung den hardwareunabhängigen Zugriff auf Interaktionsgeräte zu ermöglichen. Dieser Zugriff erfolgt über Funktionsaufrufe sowie die Erzeugung und Abfrage von von logischen Geräten. Im folgenden Abschnitt werden zunächst die verfügbaren logischen Geräteklassen präsentiert. Anschließend wird dargelegt, wie logische Geräte auf physische Geräte abgebildet werden. Schließlich wird darauf eingegangen, wie die Geräteschnittstelle mit Demon- und Serverprozessen kommuniziert und mit welchen Maßnahmen Fehler behandelt werden.

Hierarchie logischer Geräte

Wie in Forderung 1 definiert, muß ein System zur Integration multidimensionaler Interaktionsgeräte eine hardwareunabhängige logische Geräteschnitt-

stelle bieten. IDEAL definiert eine Hierarchie logischer Geräte, die in Abbildung 5.8 dargestellt ist.

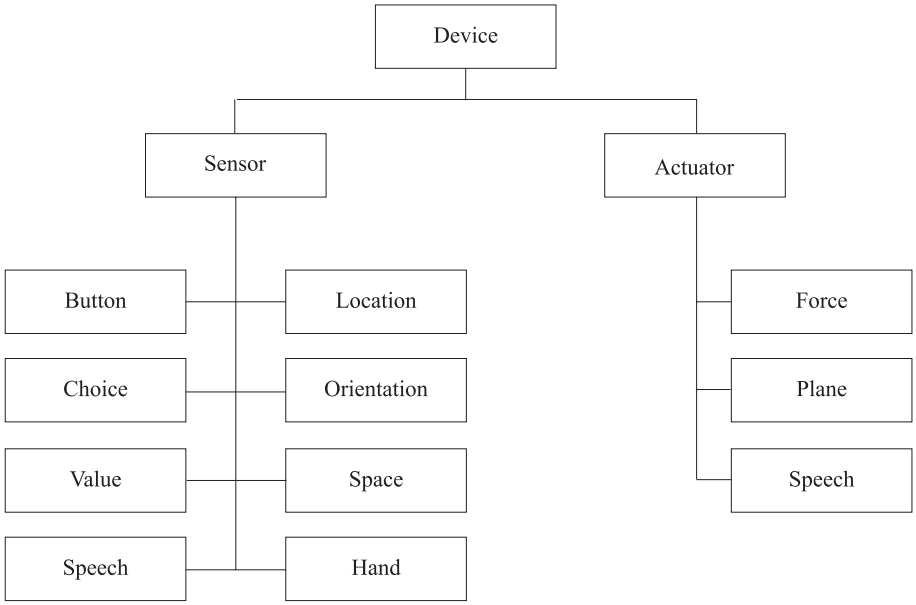

Abbildung 5.8: Hierarchie logischer Geräte in IDEAL

Die abstrakte Basisklasse aller logischen Geräte bildet *Device*. Die davon abgeleiteten Klassen *Sensor* und *Actuator* stellen ihrerseits abstrakte Basisklassen für Ein- bzw. Ausgabegeräte dar.

Sensoren

Button (Taste) Geräteabstraktion für Tasten und Knöpfe. Der Zustand eines Buttons ist entweder *an* oder *aus*. Über Attribute kann eingestellt werden, ob der Zustand des Buttons unverändert oder invertiert geliefert werden soll. Ebenso kann definiert werden, ob der Zustand des Buttons den Zustand der Taste wiedergibt, oder sich bei jeder Betätigung ändert (*toggle*).

Value (Wertgeber) Sensoren mit einem Freiheitsgrad, der in einem kontinuierlichen Wertebereich [0...1] liegt. Dieser Wertebereich kann durch Angabe eines Offsets und einer Skalierung eingestellt werden.

Choice (Auswahl) Logische Geräte dieser Klasse liefern eine "1 aus N" Auswahl. Auf Sensoren dieser Klasse können Geräte mit diskretem Wertebereich abgebildet werden.

Location (3D-Positionsgeber) Für relative Eingabegeräte wird die absolute Position akkumuliert. Standardmäßig werden Positionen in metrischen Koordinaten geliefert, es können jedoch Skalierungen und Koordinatensysteme gesetzt werden (siehe Abschnitt 5.4.7).

Orientation (3D-Orientierungsgeber) Werte werden als Rotationsmatrix geliefert. Auch hier werden die Werte relativer Eingabegeräte akkumuliert. Orientations können ebenfalls skaliert oder in ein Koordinatensystem transformiert werden.

Space Kombination aus Location und Orientation.

Hand Sensoren dieser Klasse stellen eine Beschreibung der menschlichen Hand dar. Es werden 15 Werte für die Beugung der Fingergelenke, 5 Werte für die Fingerspreizung und zwei Werte für das Handgelenk berücksichtigt. Liefert ein Datenhandschuh weniger Werte, können die Werte für die Beugung der Fingergelenke simuliert werden.

Speech (Spracheingabe) Sensor für die Spracheingabe. Der Text wird in Form eines Strings an die Applikation geliefert und kann dort, etwa durch einen Interpreter weiterverarbeitet werden.

Aktuatoren

Force (Kraft und Drehmoment) Aktuatoren dieser Art stellen bilden Kraft und Drehmoment auf ein haptisches Display ab. Diese wird in Form einer Matrix an den Aktuator übergeben.

Plane (Ebene) Es wird eine Ebene im Raum dargestellt. Die Ebene wird in Form von Basispunkt und Normalenvektor definiert. Es wird die Gegenkraft bei Berührung einer gedachten Ebene im Raum simuliert. Zusätzlich kann eine Federkonstante (stiffness) definiert werden.

Speech (Sprachausgabe) Dieser Aktuator dient zur akustischen Ausgabe von gesprochenem Text. Dieser wird in Form eines Strings an den Aktuator übergeben und mit Hilfe eines Sprachausgabesystems ausgegeben.

5.4.5 Zugriff auf logische Geräte

Im Programm erfolgt die Definition und der Zugriff auf logische Geräte über Instanzen der logischen Geräteklassen. IDEAL bietet zwei Modi zur Abfrage der Gerätedaten. Im sogenannten Pollingmodus werden die Gerätedaten bei Bedarf durch Aufruf einer Methode ausgelesen. Im Eventmodus wird dem logischen Gerät eine Callbackfunktion zugeordnet, die bei Eintreffen neuer Gerätedaten durch IDEAL aufgerufen wird. Im folgenden Beispiel wird die Verwendung von logischen Sensoren durch eine Applikation skizziert:

```
void markerMoved(IdLocation *location, void *cbd);
IdInit();
IdLocation marker(''Marker'');
marker.setCallback( markerMoved, NULL );
//Other initializations
while (1) // Main loop
{
IdPoll();
// perform simulation and render image
}
```

Zunächst erfolgt die Deklaration der Callbackfunktion. Diese enthält typischerweise Code, der die gelieferten Gerätedaten weiter verarbeitet, etwa ein grafisches Objekt bewegt. Anschließend wird die Geräteschnittstelle initialisiert. Daraufhin wird ein 3D-Positionsgeber erzeugt und benannt. Unter diesem Namen ist das Gerät nun bei IDEAL bekannt und wird einem physikalischen Gerät zugeordnet (siehe folgender Abschnitt). Mit der Zuordnung der Callbackfunktion ist das Gerät nun einsatzbereit und die Hauptschleife der Anwendung wird begonnen. Innerhalb der Hauptschleife wird zyklisch die Funktion `IdPoll` aufgerufen, um IDEAL die Möglichkeit zu geben, die Sensoren mit neuen Gerätedaten zu aktualisieren und ggf. die passenden Callbacks aufzurufen.

Konfiguration

Die Abbildung von logischen Geräten auf physikalische Geräte erfolgt nach Forderung 2 über externe Konfigurationsdateien. Damit auch Forderung 6 erfüllt werden kann, also gleichzeitig mehrere physikalische Geräte eines Typs verwendet werden können, erfolgt die Abbildung zweistufig, über eine Sensordatei, die für jede Applikation vorhanden sein muß und ein global gültiges Geräteverzeichnis.

Die erste Stufe bildet ein logisches Gerät über Vorschriften in der Sensordatei auf einen Eintrag im Geräteverzeichnis ab. Es wird definiert, von welchem Rechner die Daten angefordert werden und welcher Typ von Daten gewünscht wird. Das Geräteverzeichnis enthält für alle Rechner je einen Eintrag pro angeschlossenes Gerät. Es definiert den Namen, unter dem dieses Gerät bei IDEAL bekannt ist, an welche Schnittstelle es angeschlossen ist, sowie weitere gerätespezifische Parameter. Abbildung 5.5 illustriert den Zusammenhang zwischen Geräteverzeichnis und Sensordateien einerseits und den Systemkomponenten von IDEAL andererseits.

Beim Erzeugen eines logischen Gerätes wird zunächst die Sensordatei gelesen. Es wird nach einem zum Namen des Gerätes passenden Eintrag gesucht und dieser ausgewertet. Der zweite Schritt der Abbildung erfolgt nun unter Verwendung des Geräteverzeichnisses. Wird ein passender Eintrag gefunden, kann eine Verbindung zu diesem Gerät aufgebaut und die gewünschten Daten gelesen werde. Treten bei dieser Abbildung Fehler auf, etwa durch falsche Zuordnungsvorschriften oder weil das Gerät nicht einsatzbereit ist, wird die Initialisierung des logischen Gerätes angehalten und der Anwender durch eine entsprechende Fehlermeldung informiert. Nun ist die Gelegenheit, den Fehler in der Sensordatei oder im Geräteverzeichnis zu korrigieren oder ein anderes, einsatzbereites Gerät zu wählen und mit der Anwendung fortzufahren. Die Initialisierung erfüllt damit die Forderung 5 (Rekonfigurierbarkeit ohne Programmabbruch).

Das Gesagte soll anhand von Beispielen illustriert werden. Zunächst ein zum Codefragment auf Seite 95 passender Eintrag in die Sensordatei.

```
LINK Marker HOST rubens DEVICE Fastrak SENSOR location1
```

Der Positionsgeber `Marker` aus dem obigen Beispiel wird dem physikalischen Gerät mit Namen `Fastrak` zugeordnet, das an den Rechner `rubens` angeschlossen ist. Der Name hinter dem Schlüsselwort `SENSOR` identifiziert dabei den ersten Sensor dieses Gerätes, sowie die zu liefernden Daten (3D-Positionen). Es folgt der passende Eintrag im Geräteverzeichnis:

```
DEVICE Fastrak HOST rubens PATH /usr/ideal/FastrakServer
ARG -port=/dev/ttyd2 -baud=38400
```

Am Rechner mit Namen `rubens` ist ein Gerät angeschlossen, welches mit Hilfe des Programms `FastrakServer` über die serielle Schnittstelle Nr. 2 mit 38400 Baud angesprochen wird.

5.4.6 Kommunikation mit zwischen Geräteschnittstelle und Servern

Jede Erzeugung eines logischen Gerätes veranlaßt die Geräteschnittstelle, eine Verbindung zu einem Geräteserver aufzubauen. Wie bereits dargelegt wurde, wird über eine zweistufige Abbildung ermittelt, welcher Server angefordert werden muß auf welchem Rechner dieser läuft und welche Gerätedaten übertragen werden sollen. Die logische Geräteschnittstelle enthält dazu eine Liste aller aktiven Geräteserver und der von ihnen angeforderten logischen Geräte. Während der Ausführung der Applikation durchlaufen die Einträge in dieser Liste einen Lebenszyklus der wiederum als endlicher Zustandsautomat beschrieben werden kann, wie in Abbildung 5.9 dargestellt. Beim Auftreten eines Fehlers wird versucht, wieder einen definierten Zustand zu erreichen.

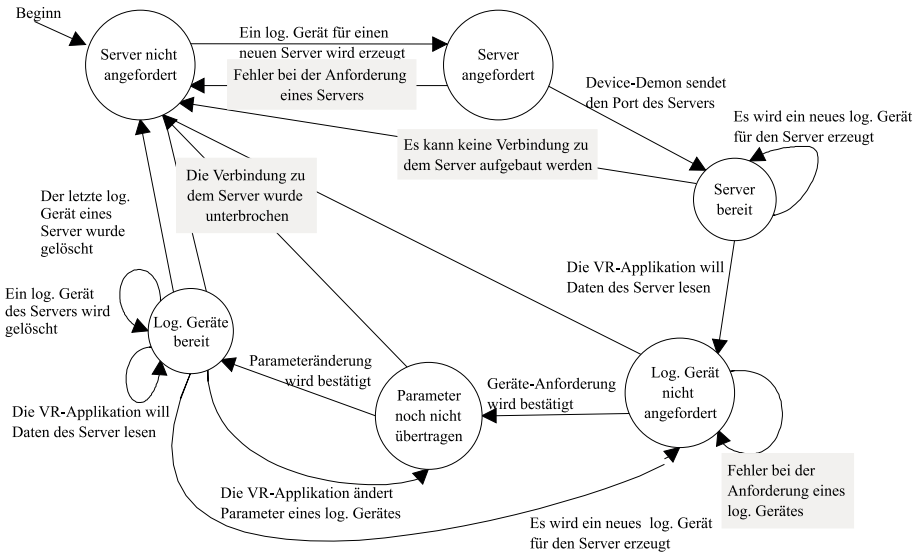

Abbildung 5.9: Zustandsübergangsdiagramm der log. Geräteschnittstelle

Wird ein logisches Gerät von einem Server angefordert, zu dem noch keine Verbindung besteht, wird zunächst beim Demon nach einer Verbindung zu diesem Server angefragt. Der Demon antwortet mit der Port Nummer, unter der der Server zu erreichen ist. Ein Eintrag für den Server wird angelegt, und diesem ein Eintrag für das angeforderte logische Gerät zugeordnet. Sobald nun die Applikation zum ersten mal Daten von diesem logischen Gerät lesen oder im Falle von Aktuatoren senden will, wird das Gerät tatsächlich beim Server angefordert, welcher diese Anforderung bestätigt. Im Fehlerfall wird der Benutzer informiert und die Anforderung kann wiederholt werden. Nun können Daten gesendet und empfangen werden. Der Zugriff auf wei-

tere logische Geräte kann von dem nun aktiven Server angefordert werden, ebenso können logische Geräte abgemeldet werden, wenn die entsprechenden Objekte von der Anwendung gelöscht werden. Wird das letzte logische Gerät welches einem Server zugeordnet war gelöscht, wird die Verbindung zu diesem Server beendet.

5.4.7 Transformation von 6D-Gerätedaten

Wie bereits in Abschnitt 5.2 ausgeführt, liefern Interaktionsgeräte Daten in herstellerspezifischen Formaten. Im Falle von 6D-Geräten existieren darüberhinaus unterschiedliche Koordinatensysteme. Damit Applikationen tatsächlich unabhängig von den verwendeten Interaktionsgeräten entwickelt werden können, bietet IDEAL Anwendungen ein einheitliches Koordinatensystem an.

IDEAL führt die Konversion innerhalb der logischen Geräte transparent für die Anwendung durch[2]. Darüberhinaus kann die Anwendung die Werte jedes Sensors nach Bedarf selbst skalieren oder beliebig transformieren.
IDEAL verwendet ein metrisches Koordinatensystem, was bedeutet, daß alle Positionswerte in Bruchteilen bzw. Vielfachen eines Meters dargestellt werden. Rotationen werden Radians $-\pi \leq \phi \leq +\pi$ geliefert.
Wie bereits in Abschnitt 5.2 ausgeführt liefern Eingabegeräte, die am Arbeitsplatz stationär aufgestellt werden *relative* Daten. In der Anwendung will man solche Geräte aber meist wie *absolute* Geräte verwenden. IDEAL behandelt daher standardmäßig alle Eingabegeräte wie absolute Geräte. Die Eingabedaten relativer Geräte werden dazu akkumuliert und diese der Applikation präsentiert. Applikationen die an den relativen Daten interessiert sind können diese anfordern, in dem sie das logische Eingabegerät in den relativen Modus versetzen. In beiden Fällen werden die relativen Daten innerhalb des Sensors zunächst in Geschwindigkeitswerte (Meter pro Sekunde) umgerechnet. Die Maximalgeschwindigkeit kann für jeden logischen Sensor von der Applikation gesetzt werden, standardmäßig ist 1 m/s vorgegeben. Auf diese Weise wird wird gewährleistet, daß physische Geräte unabhängig von der gelieferten Datenrate untereinander ausgetauscht werden können, ohne daß sich die Geschwindigkeit bei der Interaktion ändert. Für logische Eingabegeräte im absoluten Modus können für Translationen und Rotationen Minimal- und Maximalwerte definiert werden. Abbildung 5.10 stellt die Wandlung von gerätespezifischen Koordinaten in logische Koordinaten noch einmal grafisch dar.

[2]Dies geschieht, ebenso wie andere Operationen auf Gerätedaten, wie etwa Filterung oder Kalibrierung innerhalb der Serverprozesse. Damit wird die Anwendung von diesen

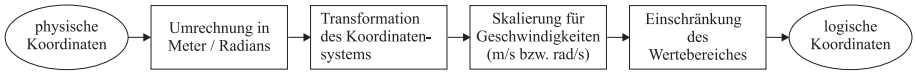

Abbildung 5.10: Transformation physischer in logische Koordinaten

5.4.8 Latenz und deren Reduktion durch optimierte Kommunikationsprotokolle

Die Latenz beschreibt den Zeitraum, der zwischen dem Auftreten eines Ereignisses und dessen Verarbeitung. In Bezug auf Interaktionsgeräte bedeutet Latenz die Zeit, die zwischen der Benutzerinteraktion bis zu deren grafischer Darstellung vergeht. Jede Komponente der Interaktionsschleife (siehe Abbildung 5.4 auf Seite 87) trägt zur beobachteten Gesamtlatenz bei. Der Einfluß der einzelnen Schritte der Interaktionsschleife soll im Folgenden betrachtet werden.

Erzeugung der physischen Gerätedaten im Interaktionsgerät

Bereits der Meßprozeß im Gerät ist latenzbehaftet. Zusätzlich arbeiten beispielsweise Trackingsysteme mit einem gewissen internen Takt bei der Meßwertaufnahme, der ebenfalls zur Latenz beiträgt. Informationen über geräteinterne Latenz sind im allgemeinen nicht verfügbar, sodas keine Aussagen über deren Anteil an der Gesamtlatenz gemacht werden können.

Übertragung der physischen Gerätedaten zum Geräteserver

Diese Übertragung erfolgt typischerweise über serielle Verbindungen. Da die Datenübertragung auf Seite des Geräteservers immer ungepuffert erfolgt, wird die Latenz bei der Übertragung durch die Baudrate und das zu übertragende Datenvolumen pro Datensatz determiniert. Betrachten wir zunächst das Datenvolumen eines typischen multidimensionalen Interaktionsgerätes. Ein elektromagnetischer Tracker mit 4 Sensoren erzeugt vier Translations- und vier Orientierungsvektoren bestehend aus je drei Koordinaten, insgesamt also 24 Gleitkommawerte.

Die meisten Geräte bieten zwei Modi für Übertragung von Gleitkommawerten, Binär und Ascii. Im Binärmodus werden die Zahlen als Bitmuster übertragen, im Asciimodus muß für jede Ziffer ein Zeichen übertragen werden. Für eine 32-Bit Gleitkommazahl müssen im Binärmodus 4 Byte übertragen werden, im Asciimodus 8 Byte. Ein Datensatz im Binärmodus ist also

Operationen entlastet.

halb so groß wie im Asciimodus. Der oben genannte Datensatz besteht also aus 96 Bytes im Binärmodus und 192 Bytes im Asciimodus.

Die Baudrate für serielle Schnittstellen von Interaktionsgeräten liegt zwischen 9600 Baud (Spacemouse) bis zu 115200 Baud (Polhemus Fastrak) oder mehr. Im ungünstigsten Fall (9600 Baud, 192 Bytes pro Datensatz entsteht eine Latenz von 160 Millisekunden, im besten Fall (115200 Baud, 96 Bytes) beträgt die Latenz 6.7 Millisekunden[3]).

Zusammenfassend kann gesagt werden, daß die Latenz durch die serielle Verbindung einen nicht zu vernachlässigenden Anteil an der Gesamtlatenz hat. Wo immer möglich sollte binär bei möglichst hoher Baudrate übertragen werden.

Verarbeitung innerhalb des Geräteservers

Latenz entsteht hier hauptsächlich durch Filteralgorithmen, die zur Datenglättung eingesetzt werden. Diese berücksichtigen typischerweise 3-20 Datenwerte (vergl. [Felg95] Seite 108). Die Anzahl der bei der Filterung berücksichtigen Datenwerte wirkt sich direkt multiplikativ auf die bisher akkumulierte Latenz aus. Hier muß eine Abwägung zwischen der benötigten Datenglättung und der in Kauf zu nehmenden Latenz getroffen werden.

Latenz des Netzwerkes

Bei der Datenübertragung über ein lokales Kommunikationsnetzwerk entsteht ebenfalls Latenz im Bereich mehrerer Millisekunden. Laufen Geräteserver und Anwendung dagegen auf dem selben Rechner, bleibt die Latenz vernachlässigbar. Um genauere Informationen über den Einfluß der Netzwerklatenz zu erhalten, wurden Messungen durchgeführt. Das in Abbildung 5.11 dargestellte Diagramm stellt die Latenz in Abhängigkeit von der Größe des angeforderten Datensatzes dar. Untersucht wurden zwei unterschiedliche Protokolle, das verbindungsorientierte TCP/IP und das verbindungslose UDP/IP. Die beiden oberen Kurven wurden gemessen, wenn Anwendung und Geräteserver auf verschiedenen Rechnern laufen, die unteren Kurven sind entstanden, als beide auf demselben Rechner liefen. Wie zu erwarten, steigt die Latenz mit der Länge des angeforderten Datensatzes. Ebenso konnte erwartet werden, daß das verbindungslose Protokoll UDP/IP weniger latenzbehaftet sein würde, als das wesentlich aufwendigere TCP/IP. Allerdings fiel der Unterschied so gering aus, daß das einfacher zu handhabende Protokoll TCP/IP für die Datenübertragung gewählt wurde.

[3]In Wirklichkeit würden die Ergebnisse schlechter ausfallen, da der Protokolloverhead hier vernachlässigt wurde

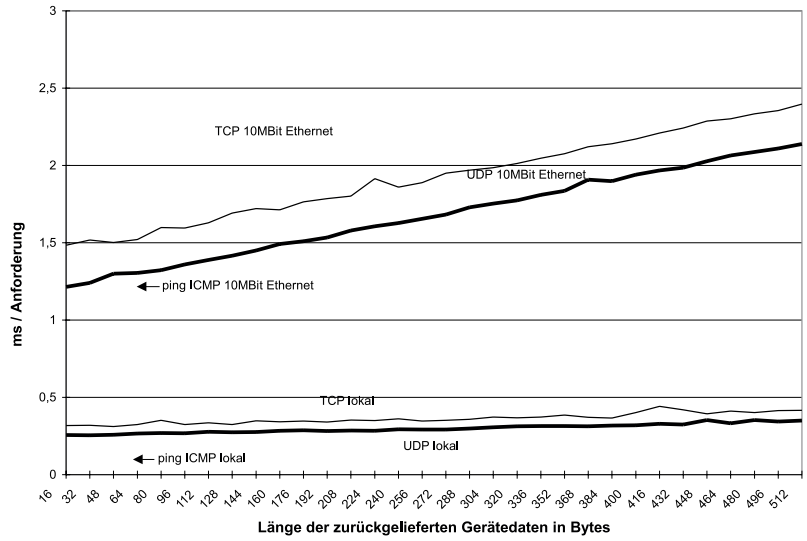

Abbildung 5.11: Netzwerklatenz

Latenz innerhalb der Applikation

Die Latenz innerhalb der Anwendung wird hauptsächlich durch die Zykluszeit der Hauptschleife bestimmt. Diese wiederum hängt bei grafisch-interaktiven Anwendungen hauptsächlich von der Geschwindigkeit der Bildgenerierung (Rendering Framerate) ab.

Reduktion der Netzwerklatenz durch optimierte Protokolle

Die vorangegangenen Betrachtungen haben gezeigt, daß Latenz in jeder Stufe der Interaktionsschleife entsteht. Die ersten beiden Stufen sind durch die Gerätehardware bedingt und nur durch optimale Parametrierung beeinflußbar. Innerhalb der Geräteserver entsteht Latenz dann, wenn Daten durch Filterung geglättet werden müssen. Die darauf folgenden Stufen der Interaktionsschleife erlauben aber eine Reduktion der Latenz durch optimierte Kommunikationsprotokolle.

Betrachten wir zunächst das Kommunikationsprotokoll in Abbildung 5.12.

Die Applikation fordert einen Wert vom Geräteserver an und wartet, bis dieser eintrifft. Anschließend wird ein neues Bild berechnet. Auch wenn dieser Ansatz naiv erscheinen mag, verursacht er doch die minimale Latenz, die sich wie folgt berechnet: $t_{latenz} = t_{trans} + t_{frame}$ Hierbei definiert t_{trans} die Netzwerklatenz und t_{frame} die Dauer der Bildgenerierung durch die Anwendung. Inakzeptabel ist aber die Tatsache, daß die Anwendung während der

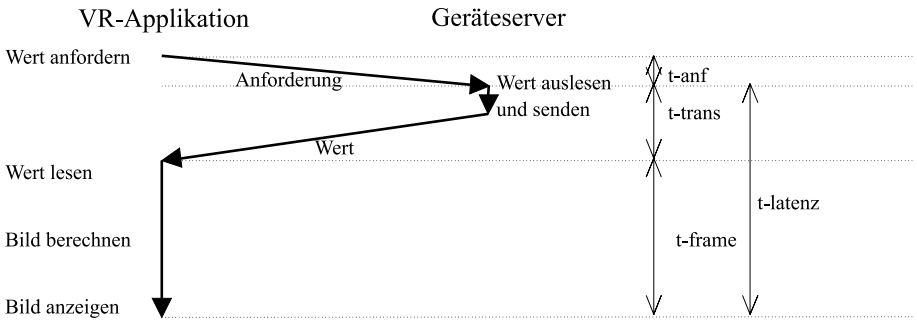

Abbildung 5.12: Synchrones Kommunikationsprotokoll

Anforderung der Daten untätig warten muß.

Der zweite Ansatz (Abbildung 5.13) wird von allen untersuchten Systemen realisiert und kommt ohne Wartezeit während der Bereitstellung der Gerätedaten aus.

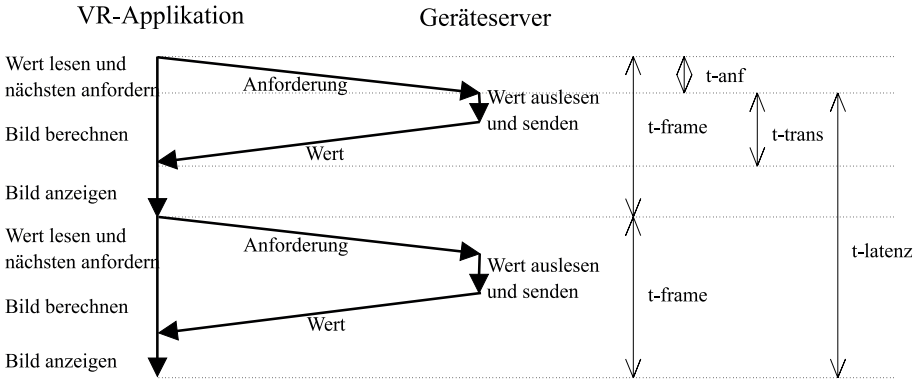

Abbildung 5.13: Asynchrones Kommunikationsprotokoll

Es werden zunächst Gerätedaten angefordert, die Anwendung wartet aber nicht auf deren Eintreffen sondern berechnet sofort ein neues Bild.Inzwischen treffen neue Gerätedaten ein, die für die Berechnung des nächsten Bildes verwendet werden können. Die Latenz ergibt sich nach folgender Formel: $t_{latenz} = 2t_{frame} - t_{anf}$. In typischen Anwendungen innerhalb eines lokalen Netzwerkes beträgt erfolgt die Reaktion auf die Benutzerinteraktion zwei Bilder zu spät.

Das in Abbildung 5.14 dargestellte Protokoll verzichtet auf Synchronisierung zwischen Anwendung und Geräteserver.

Der Geräteserver sendet Daten in der maximal möglichen Geschwindigkeit. Die Applikation verwirft alle Werte bis auf den zuletzt empfangenen. Die Latenz ergibt sich zu $t_{trans} + t_{frame} \leq t_{latenz} \leq 2t_{frame}$. Dieser Wert liegt

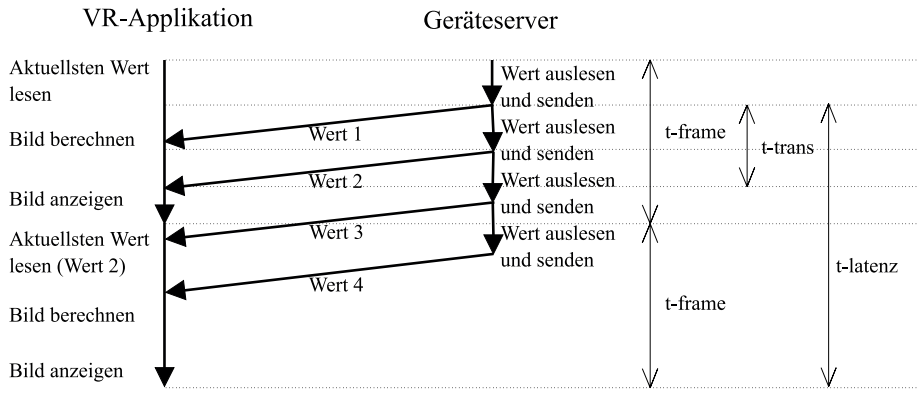

Abbildung 5.14: Unsynchronisiertes Kommunikationsprotokoll

zwischen dem ersten und zweiten Ansatz. Die Flut der ungenutzten Daten verschwendet aber Netzwerkbandbreite und Rechenzeit auf Seite der Server.

IDEAL verwendet ein erweitertes asynchrones Protokoll, daß in Abbildung 5.15 gezeigt ist.

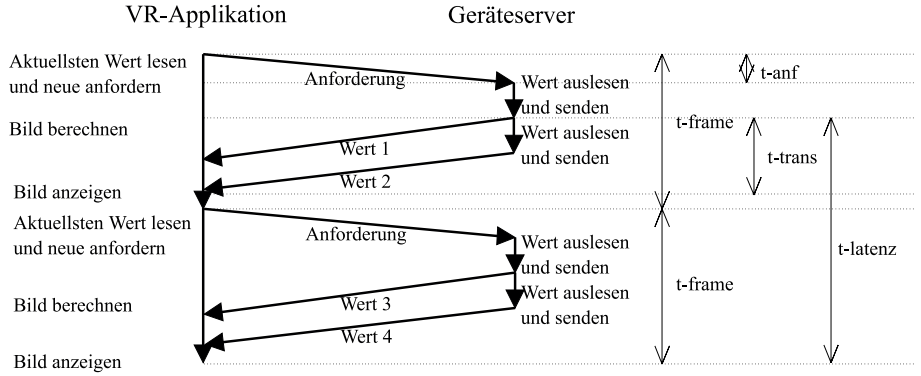

Abbildung 5.15: Optimales Kommunikationsprotokoll

Die Anwendung fordert Daten vom Geräteserver an und beginnt sofort ein neues Bild zu generieren. Der Geräteserver liest die Daten und sendet sie sofort an die Applikation zurück. Sind neue Gerätedaten verfügbar, bevor eine neue Anforderung vorliegt, werden diese ebenfalls sofort gesendet. Die Zahl der unaufgefordert gesendeten Werte wird auf eine kleine Zahl begrenzt. Die Latenz kann wie folgt abgeschätzt werden: $t_{trans} + t_{frame} \leq t_{latenz} \leq 2t_{frame} - t_{trans}$. IDEAL erreicht so die kleinste Latenz aller Protokolle, die nicht zur Blockierung der Anwendung führen.

5.4.9 Die grafische Benutzungsoberfläche

Wie in Abschnitt 5.4.5 dargestellt, wird IDEAL durch zwei Dateien konfiguriert. Die Sensordatei für jede Anwendung und das Geräteverzeichnis. Für eine korrekte Einstellung ist jedoch einiges Fachwissen über die verfügbaren logischen und physischen Geräte erforderlich. Ein Schreibfehler in einer der beiden Dateien führt zu Fehlern bei der Initialisierung von Geräten oder der Anwendung und muß zunächst korrigiert werden. Auch ist es wünschenswert, einzelne Geräte auf korrekte Funktion zu überprüfen, ohne dazu eine komplexe Anwendung zu starten. Schließlich benötigen einige Interaktionsgeräte eine aufwendige Kalibrierung, für deren fehlerfreie Durchführung wieder Fachwissen erforderlich ist. Daher lag es nahe, die beschriebenen Vorgänge in einem Programm zusammenzufassen und mit einer grafischen Benutzungsoberfläche zu versehen. Diese sollte die Bedienung von IDEAL weitestmöglich vereinfachen, Plausibilitätskontrollen durchführen und Hilfen anbieten.

Die grafische Oberfläche von IDEAL , IdAdmin leistet dies. IdAdmin besteht aus zwei Hauptkomponenten, dem Sensor- und dem Gerätebrowser. Diese werden in den nächsten Abschnitten vorgestellt. Anschließend werden die Test- und Kalibrierungswerkzeuge vorgestellt.

Sensorbrowser

Aufgabe des Sensorbrowsers ist es die Abbildung von logischen auf physische Geräte zu erleichtern. Bestehende Sensordateien können bearbeitet oder neu erzeugt werden. Seine Benutzungsoberfläche ist in Abbildung 5.16 dargestellt.

Auf der linken Seite der Abbildung ist das Verzeichnis aller verfügbaren Interaktionsgeräte zu sehen, in diesem Fall nach Gerätetypen sortiert. Für jeden Typ kann überblickt werden, an welchen Rechnern ein solches Gerät verfügbar ist. Die rechte Seite der Abbildung zeigt eine Sensordatei, die Einträge für sieben logische Geräte enthält. Jede Zeile stellt eine Abbildungsvorschrift dar, die vierte wird im Moment editiert. Das geöffnete Menü zeigt eine Liste von verfügbaren Geräten an, die dem gewählten logischen Gerät zugeordnet werden können.

Gerätebrowser und Dialoge

Die Aufgabe des Gerätebrowsers ist die Pflege des Geräteverzeichnisses. Dieser besteht aus zwei Teilen. Auf der linken Seite von Abbildung 5.17 ist das Verzeichnis aller verfügbaren Geräte dargestellt. Dieses kann entweder nach Rechnern oder Gerätetypen sortiert werden. Die Abbildung zeigt die Sortierung nach Rechnern. Dabei werden die Geräte in Form einer Hierarchie unterhalb der Rechner dargestellt, an die diese angeschlossen sind.

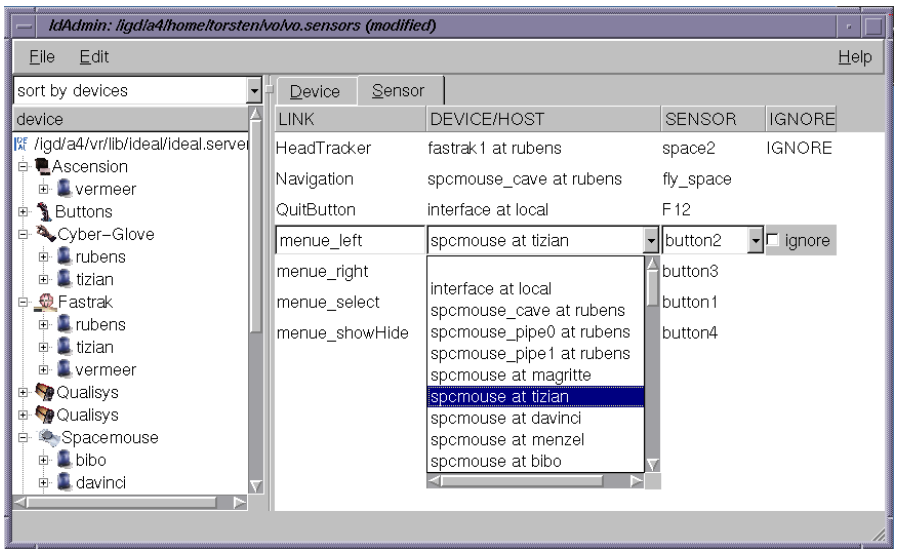

Abbildung 5.16: Benutzungsoberfläche des Sensorbrowsers

Die rechte Seite von Abbildung 5.17 zeigt den Gerätedialog des aktuell ausgewählten Gerätes, hier eines Datenhandschuhs. Der obere Teil des Gerätedialoges faßt Parameter zusammen, die für alle Geräte gesetzt werden müssen. Diese umfassen den symbolischen Namen des Gerätes, den Rechner und die Schnittstelle an die das Gerät angeschlossen ist, den passenden Geräteserver und die Protokolldatei.

Im umrahmten mittleren Teil finden sich die gerätespezifischen Parameter, wie Kalibrierungsdateien, Baudrate, Filterparameter etc.

Die Knöpfe im unteren Teil erlauben die Rücknahme von Eingaben, sowie den Aufruf von Test- und Kalibrierungsdialogen. Darunter findet sich ein Feld für Kommentare und Informationen zu dem Gerät.

Die Konfiguration des Gerätes wird vom Gerätedialog bereits bei der Eingabe auf Plausibilität und korrekte Syntax überprüft. Etwa werden sinnvolle Werte für die Baudrate oder Dateinamen gefordert und die Eingabe andernfalls abgelehnt. So wird gewährleistet, daß das Geräteverzeichnis unter allen Umständen syntaktisch korrekt bleibt. Die Wahl der Parameter kann durch Betätigung des Test-Knopfes sofort überprüft werden.

Lineare Trackerkalibrierung

Für Trackingsysteme aller Art muß eine lineare Kalibrierung durchgeführt werden, bevor diese sinnvoll eingesetzt werden können. Dabei wird das Gerätekoordinatensystem in das einheitliche logische Koordinatensystem (siehe Ab-

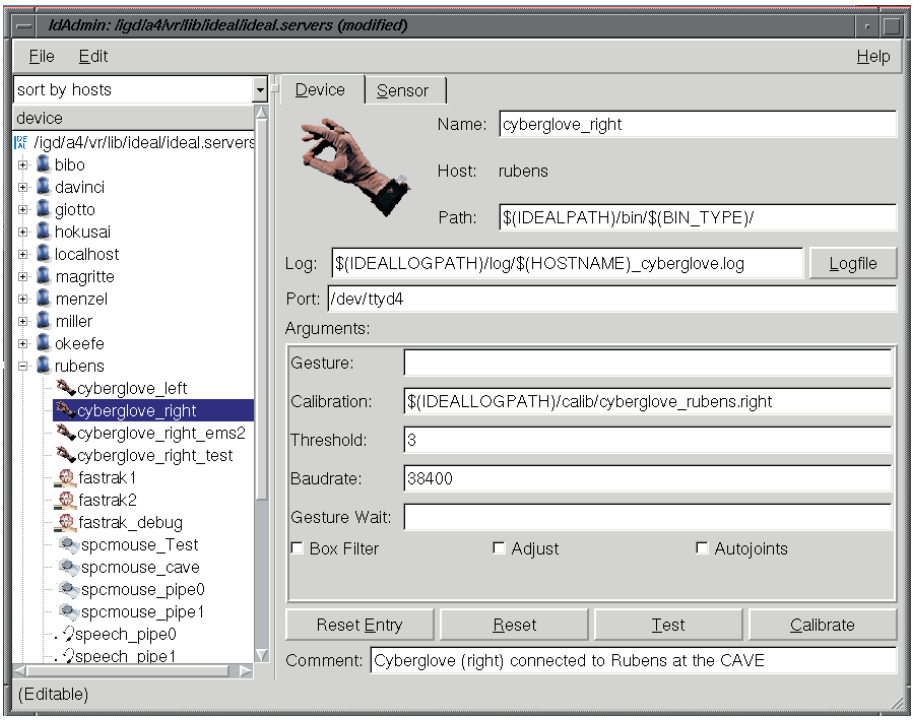

Abbildung 5.17: Benutzungsoberfläche des Gerätebrowsers

schnitt 5.4.7) von IDEAL abgebildet.

Diese Kalibrierung besteht aus zwei Stufen. Zunächst wird das Basiskoordinatensystem bestimmt. Dazu wird ein Sensor des Trackingsystems in die gewünschte Ursprungsposition gebracht und diese eingemessen. Es wird eine Reihe von Werten gelesen um Zittern zu unterdrücken. Der Sensor muß dabei so gehalten oder befestigt werden, daß seine Vorderseite in Richtung der für die aktuelle Installation sinnvolle Vorwärtsrichtung zeigt, etwa in Richtung der Leinwand oder des Bildschirms. Die Oberseite des Sensors muß nach oben zeigen[4].

Im zweiten Schritt kann eine zusätzliche Transformation für jeden einzelnen Sensor definiert werden. Dies ist dann sinnvoll, wenn die Sensoren an Geräten befestigt sind und ihre Lage nicht dem gewünschten Referenzpunkt entspricht. Beispielsweise sei ein Tracker an der linken Seite einer Stereobrille senkrecht befestigt. Würde die Position des Sensors direkt eine virtuelle Kamera steuern, würde diese, wenn die Brille waagrecht gehalten wird nach

[4]In einigen Fällen ist es schwierig zu erkennen, welches die Vorderseite und Oberseite eines Sensors sind, etwa bei optischen Trackingsystemen, bei denen Konstruktionen aus mehreren kugelförmigen Markern verwendet werden. Hier kann eine Festlegung oft nur willkürlich getroffen werden.

oben sehen. Auch liegt der gedachte Blickpunkt der Kamera nicht an der linken Seite der Brille, sondern in deren Mitte zwischen den Augen. Entsprechendes gilt für Datenhandschuhe oder Werkzeuge, die in der Hand gehalten werden. Abbildung 5.18 zeigt den linearen Kalibrierungsdialog.

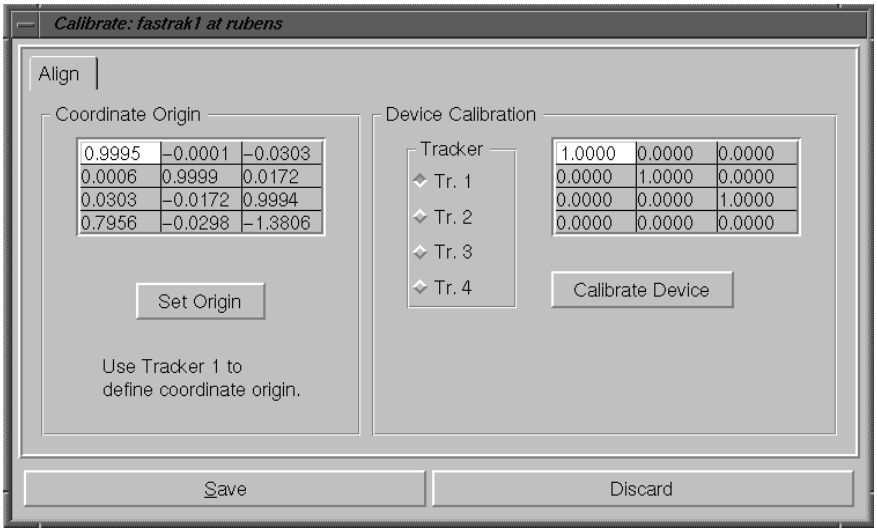

Abbildung 5.18: Dialog für die lineare Trackerkalibrierung

Nichtlineare Trackerkalibrierung

Elektromagnetische Trackingsysteme werden durch das Vorhandensein elektrischer Geräte oder metallischer Gegenstände in der Umgebung gestört. Diese verzerren das elektromagnetische Feld, welches Trackingsysteme aufbauen und das von den Trackingsensoren gemessen wird, um Position und Orientierung zu bestimmen. Diese Störungen führen zu Ungenauigkeiten, die so groß werden können, daß die gemessenen Werte schlicht unbrauchbar werden [Zach97].

Werden die störenden Gegenstände nicht bewegt, ist die Verzerrung des Magnetfeldes konstant und kann gemessen werden. Aus einer Reihe von Meßwerten kann man anschließend eine nichtlineare Verzerrungsfunktion im Raum berechnen. Diese ordnet jeder Raumkoordinate eine verzerrte Koordinate zu. Die Umkehrfunktion dieser Verzerrungsfunktion kann man nun als Entzerrungsfunktion auf die gemessenen Werte anwenden. Dadurch kann die Verformung des Magnetfeldes ausgeglichen werden.

IDEAL bietet dazu einen Dialog für nichtlineare Trackerkalibrierung, der den Benutzer bei der Meßwertaufnahme unterstützt und die Entzerrungs-

funktion berechnet. Die durch die bisher aufgenommenen Meßwerte ermittelte Magnetfeldentzerrung kann jederzeit grafisch dargestellt und überprüft werden. Abbildung 5.19 zeigt den Dialog zur Entzerrung des Magnetfeldes.

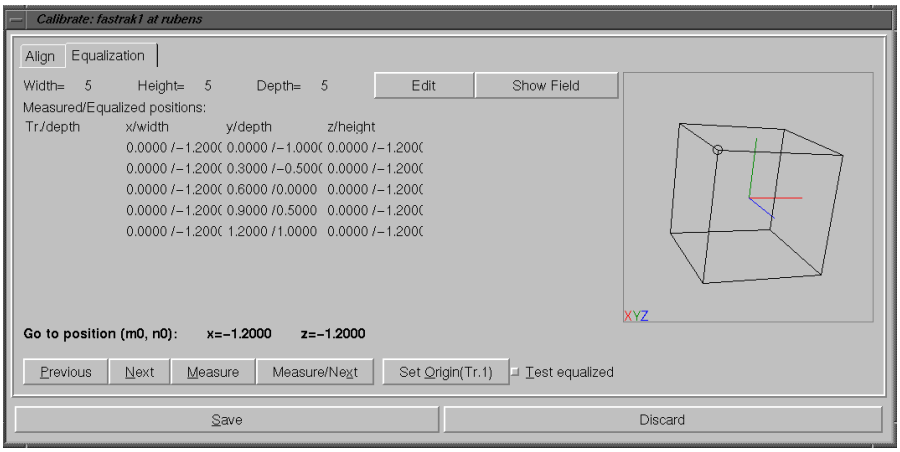

Abbildung 5.19: Dialog zur Magnetfeldentzerrung

Kalibrierung von Datenhandschuhen

Datenhandschuhe müssen kalibriert werden, um die von den Beugungssensoren gelieferten Werte korrekt auf Gelenkwinkel abzubilden. Dies ist besonders dann wichtig, wenn Objekte in Virtuellen Umgebungen präzise manipuliert werden sollen. Auch für die Gestenerkennung ist eine saubere Kalibrierung des Datenhandschuhs wichtig.

Die Kalibrierung des Datenhandschuhs erfolgt durch eine Reihe von sechs Gesten, die vom Benutzer nacheinander durchgeführt werden müssen. Jede dieser Gesten erzeugt bei bestimmten Beugungssensoren Minimal- oder Maximalwerte. Für jede Geste werden die Sensorwerte mehrmals gelesen, um Rauschen zu unterdrücken. Der Benutzer wird Schritt für Schritt durch den Kalibrierungsvorgang geführt. Dabei ist Vor- und Zurückspringen möglich, sodas Fehler nachträglich korrigiert werden können. Ein Beispiel zeigt Abbildung 5.20.

Ist der Kalibrierungsvorgang abgeschlossen, kann das Ergebnis sofort mit Hilfe des Testdialoges überprüft werden. Dabei stehen grafische und numerische Darstellung der Gelenkwerte zur Verfügung.

Abbildung 5.20: Kalibrierungsdialog für einen Datenhandschuh

5.5 Zusammenfassung

Das im Rahmen dieser Arbeit entwickelte System IDEAL erlaubt die Integration multidimensionaler Interaktionsgeräte in grafisch-interaktive Anwendungen. Durch eine Hierarchie logischer Eingabegeräte kann die Integration hardwareunabhängig erfolgen. IDEAL umfaßt ein breites Spektrum von Interaktionsgeräten sowohl zu Eingabe als auch zur haptischen Ausgabe. Es ermöglicht den netzwerkweiten Zugriff und die fehlertolerante Handhabung dieser Geräte. Seine grafische Benutzungsoberfläche erlaubt auch weniger versierten Benutzern Parametrierung, Kalibrierung und Test dieser oftmals komplexen Systeme. Zum Zeitpunkt der Erstellung dieser Arbeit werden 10 Systeme unterstützt, zusätzlich Tastatur und Maus. Anbindungen für weitere Geräte sind in der Entwicklung. IDEAL ist seit 1998 im operativen Einsatz und Teil des kommerziellen VR-Systems Virtual Design II.