DIPLOMARBEIT

Medizin zum Anfassen

Konzeption und Entwicklung einer Augmented-Reality Umgebung

Medizin zum Anfassen

Konzeption und Entwicklung einer Augmented-Reality Umgebung

Diplomarbeit

Zur Erlangung des akademischen Grades Diplom-Informatiker (FH)

im Fachbereich Informatik und Medien

der Fachhochschule Brandenburg.

von:	Kai Frentzel
geboren am:	16.09.1979
in:	Magdeburg
Matrikelnummer:	20032069

1. Gutachter:	Prof. Dr.-Ing. Jörg Berdux
2. Gutachter:	Prof. Dr.-Ing. Thomas Preuß

Zeitraum der Diplomarbeit:	09.07.2007 - 09.10.2007

Selbständigkeitserklärung

Hiermit versichere ich, Kai Frentzel, die vorliegende Arbeit allein verfasst zu haben. Es wurden ausschließlich die in der Arbeit benannten Quellen und Hilfsmittel benutzt.

Die Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Magdeburg, den 09. Oktober 2007

Kai Frentzel

Danksagung

Mein Dank gilt an erster Stelle meiner Verlobten Katja Rausch sowie meiner Tochter Karolina. Die mir beide in den letzten Monaten den Rücken freigehalten, so manches Chaos ertragen und mir immer wieder neue Kraft gegeben haben. Meinen Betreuer Prof. Dr.-Ing. Jörg Berdux danke ich für eine hervorragende rund um die Uhr Betreuung. Er förderte durch interessante Diskussionen und Gespräche ebenfalls die Fertigstellung der vorliegenden Arbeit. Konrad Mühler möchte ich für seine zahlreichen Hinweise zum Schreiben wissenschaftlicher Arbeiten danken. Nicht zuletzt danke ich besonders meinen Eltern Irina und Ralf Frentzel für ihre Unterstützung in den letzten Jahren.

Inhaltsverzeichnis

Inhaltsverzeichnis ..... **iii**

Abkürzungsverzeichnis ..... **vii**

Kapitel 1 Einleitung **1**

1.1 Motivation und Aufgabenstellung 1

1.2 Ergebnisse 1

1.3 Gliederung 2

Kapitel 2 Grundlagen **3**

2.1 Der Begriff Augmented-Reality 3

2.2 Möglichkeiten des Trackings 4

2.2.1 Zeit- und Frequenzmessung 5

2.2.2 Mechanische Kopplungen 6

2.2.3 Inertialsysteme 7

2.2.4 Räumliche Scans 7

2.2.5 Direkte Feldmessung 8

2.2.6 Hybrid-Systeme 9

2.2.7 Zusammenfassung 10

2.3 Darstellungssysteme 11

2.3.1 Head-Mounted-Displays 11

2.3.2 Projektorbasierte Systeme 13

2.3.3 Handheld Systeme 14

2.4 Interaktionsgeräte 14

2.4.1 Datenhandschuh 14

2.4.2 3D-Maus 15

2.4.3 Pen & Tablet 15

2.5 Inhaltsverwaltung 16

2.6 Die verschiedenen Koordinatensysteme 17

2.7 Struktur einer Augmented-Reality Anwendung 19

2.8 Beschreibung des Trackingvorganges am Beispiel des ARToolKits 20

2.8.1 Vorraussetzungen ... 20
2.8.2 Erkennen eines normalen Markers ... 20
2.8.3 Erkennen eines Multimarkers ... 22
2.8.4 Grenzen des Trackingverfahrens ... 23

Kapitel 3 Augmented-Reality in Industrie, Forschung und Medizin ... 25
3.1 Augmented-Reality im industriellen Umfeld ... 25
3.2 Augmented-Reality in der Medizin ... 26
3.2.1 Intraoperative Unterstützung ... 26
3.2.2 Chirurgisches Training ... 28
3.2.3 Präoperative Planung ... 28
3.3 Augmented-Reality in anderen Bereichen ... 28
3.4 Beispiele für Augmented-Reality Anwendungen und Projekte ... 30
3.4.1 ARVIKA ... 30
3.4.2 ARTESAS ... 30
3.4.3 Liver surgery planning system ... 31
3.4.4 MEDical Augmented-Reality for PAtients ... 32
3.5 Zusammenfassung ... 33

Kapitel 4 Frameworks und verwendete Techniken ... 35
4.1 Bildaufnahme und Marker Erkennung ... 35
4.1.1 ARToolKit ... 35
4.1.2 ARToolKitPlus ... 36
4.1.3 ARTag ... 36
4.1.4 Distributed Wearable Augmented-Reality Framework ... 37
4.1.5 Mixed Reality Toolkit ... 37
4.1.6 Das OpenIllusionist Projekt ... 37
4.1.7 Zusammenfassung ... 38
4.2 Darstellung der virtuellen Objekte ... 39
4.2.1 JPCT ... 39
4.2.2 jMonkeyEngine ... 39
4.2.3 Java3D ... 40
4.2.4 Zusammenfassung ... 40
4.3 XML ... 40
4.3.1 Validierung von XML-Daten ... 40
4.3.2 XML-Parser ... 41
4.4 Weitere genutzte Techniken ... 42

Kapitel 5 Konzept einer modularen Augmented-Reality Anwendung 43
5.1 Analyse 43
5.1.1 Problembeschreibung 43
5.1.2 Mögliche Lösungsansätze 44
5.1.3 Funktionale Anforderungen 45
5.1.4 Nicht funktionale Anforderungen 45
5.2 Architektur 46
5.2.1 Komponenten 46
5.2.2 Schichten 48
5.3 Die Daten-Objekte des Frameworks 49
5.3.1 Das PatternObject-Objekt 49
5.3.2 Das VirtualObject-Objekt 50
5.4 Schnittstellen der Komponenten 51
5.4.1 Die Schnittstelle ITracking 51
5.4.2 Die Schnittstelle IManagement 53
5.4.3 Die Schnittstelle IInteraction 53
5.4.4 Die Schnittstelle IDisplay 54

Kapitel 6 Implementierung von ARMediView 55
6.1 Der ARMediView-Zeiger 55
6.2 Die Kernkomponente von ARMediView 56
6.3 Die ARMediView-Objekte 57
6.3.1 Das Pattern Objekt 57
6.3.2 Das MultiPattern Objekt 57
6.3.3 Das ARMVObject Objekt 58
6.3.4 Das Pointer Objekt 59
6.4 Implementierung der Komponenten 59
6.4.1 ARMediView - Tracking 59
6.4.2 ARMediView – Management 62
6.4.3 ARMediView – Interaction 65
6.4.4 ARMediView – Display 66

Kapitel 7 Zusammenfassung und Ausblick 69
7.1 ARMediView als Framework 69
7.2 ARMediView als Anwendung 69

Literaturverzeichnis 71

Abbildungsverzeichnis **75**

Tabellenverzeichnis **77**

Anhang A Einstieg in die Entwicklung von Augmented-Reality Anwendungen mit Hilfe des jAR-Frameworks **79**

Anhang B Quelltexte und JavaDoc **79**

Abkürzungsverzeichnis

2D	Zweidimensional
3D	Dreidimensional
API	Application Programming Interface
AR	Augmented Reality
BMBF	Bundesministerium für Bildung und Forschung
CT	Computertomographie
DOF	Degrees of Freedom - Freiheitsgrade
DOM	Document Object Model
DTD	Document Type Definition
GIS	Geoinformationssystem
GPL	General Public License
GUI	Graphical User Interface
HITLab	Human Interface Technology Lab
HMD	Head-Mounted-Display
JNI	Java Native Interface
LSPS	Liver Surgery Planning System
LWJGL	Lightweight Java Game Library
MEDARPA	MEDical Augmented Reality for PAtients
MRT	Magnetresonanztomographie
OpenGL	Open Graphics Library
OpenSG	Open Scenegraph
RWTH	Rheinisch-Westfälische Technische Hochschule
SAX	Simple API for XML
TGA	Targa Image File
USB	Universal Serial Bus
VR	Virtual Reality (virtuelle Realität)
VRD	Virtual-Retinal-Display
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consortium
XML	eXtended Markup Language

Kapitel 1 Einleitung

Diese Arbeit beschäftigt sich mit der Konzeption und Implementierung einer Augmented-Reality (Erweiterte Realität) Anwendung zur Darstellung von medizinischen Daten. Das Hauptaugenmerk wird hierbei auf das Zusammenspiel der verschiedensten Techniken gelegt. Damit wird eine Grundlage und ein Einstiegspunkt für die spätere Entwicklung weiterer Augmented-Reality Anwendungen geschaffen.

1.1 Motivation und Aufgabenstellung

„Wir leben im Informationszeitalter und merken es daran, daß wir uns vor Information nicht mehr retten können. Nicht der überwältigende Nutzen der Information, sondern ihre nicht mehr zu bewältigende Flut charakterisiert die Epoche." [Fra98, S. 49]

Dieses Zitat von Georg Franck zeigt, wie wichtig und schwierig es heutzutage ist, einen Weg zu finden, um die Menge an Informationen, die einem zur Verfügung steht, möglichst effizient zu verwalten, zu filtern und darzustellen, um bei Bedarf genau dann die richtige Information parat zu haben, die man benötigt, und nicht in einer Flut von nicht benötigten Informationen unter zu gehen.

Die Aufgabe der vorliegenden Arbeit besteht in der Konzeption und Implementierung einer Augmented-Reality Anwendung zur Darstellung von medizinischen 3D-Objekten. Der Benutzer soll dabei die Möglichkeit haben, mit diesen Objekten zu interagieren. Dies beinhaltet unter anderem das dynamische Laden von Objekten, deren Darstellung und Transformation. Das Konzept dafür wurde im Rahmen dieser Arbeit entworfen und implementiert. Dafür wurden verschiedenste Techniken miteinander kombiniert.

1.2 Ergebnisse

Im Rahmen dieser Arbeit wurde ein Konzept für eine Augmented-Reality Anwendung zur Darstellung virtueller 3D-Objekte entwickelt. Das Konzept wurde im Rahmen einer Anwendung zur Darstellung medizinischer 3D-Objekte implementiert. Mit dieser Anwendung ist es möglich, 3D-Objekte zu laden und zu betrachten. Das Konzept kann als Grundlage zur Erstellung eigener Augmented-Reality Anwendungen genutzt werden.

Weiterhin wurden Techniken für Augmented-Reality Systeme, sowie Anwendungsmöglichkeiten für Augmented-Reality vorgestellt.

1.3 Gliederung

Die vorliegende Arbeit gliedert sich wie folgt:

In **Kapitel 2** wird eine Einführung in die Grundlagen von Augmented-Reality gegeben. Neben der Erklärung des Begriffes Augmented-Reality, werden die einzelnen Komponenten einer Augmented-Reality Anwendung genauer umrissen. Weiterhin wird die Struktur einer solchen Anwendung dargestellt, sowie der Trackingvorgang anhand eines markerbasierten Traking-systems erläutert.

Einsatzgebiete für Augmented-Reality Anwendungen sowie einige Arbeiten und Projekte werden in **Kapitel 3** beschrieben. Besonderes Augenmerk wird dabei auf den medizinischen Bereich gelegt.

Kapitel 4 widmet sich der in dieser Arbeit benutzen Frameworks und Techniken. Neben der Analyse und Beschreibung verschiedener Frameworks für die Markererkennung, sowie die Erstellung und Darstellung von virtuellen Objekten, werden weitere benutzte Techniken vorgestellt.

Das in dieser Arbeit entwickelte Konzept wird in **Kapitel 5** vorgestellt. Dabei wird zunächst eine Problemanalyse durchgeführt. Anschließend wird die Architektur der Anwendung vorgestellt und die Schnittstellen der einzelnen Komponenten für das System.

Die Implementation dieses Konzeptes wird in **Kapitel 6** näher erläutert. Vom Controller der Anwendung über die benutzen Objekte bis zur Implementation der Komponenten.

In **Kapitel 7** werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick über weitere Anwendungsmöglichkeiten und Erweiterungen der entwickelten Anwendung gegeben. Im Anhang befindet sich ein Leitfaden zur Erstellung einer eigenen Augmented-Reality Anwendung auf Grundlage dieser Arbeit.

Kapitel 2 Grundlagen

2.1 Der Begriff Augmented-Reality

Augmented-Reality (erweiterte Realität[1], kurz: AR) ist eine Form der Mensch-Maschine-Interaktion und bezeichnet im Allgemeinen die Überlagerung menschlicher Sinneswahrnehmungen mit Computermodellen in Echtzeit [MC99]. Ein Augmented-Reality System kann demensprechend visuelle, akustische und haptische Informationen in Echtzeit überlagern und wiedergeben. In dieser Arbeit wird im Zusammenhang mit Augmented-Reality nicht weiter auf die Überlagerung von akustischen und haptischen Reizen sondern nur auf die Überlagerung visueller Reize eingegangen. AZUMA beschreibt in [Azu97] folgende drei charakteristische Merkmale eines AR-Systems:

- Kombination von realen und virtuellen Objekten
- Interaktivität und Echtzeitfähigkeit
- Registrierung[2] in 3D

Abgrenzung zur virtuellen Realität

Augmented-Reality ist verwandt mit der virtuellen Realität. Im Gegensatz zur virtuellen Realität (Virtual Reality, kurz VR), wo die komplette Darstellung der Umgebung und ihrer physikalischen Eigenschaften in Echtzeit durch einen Computer generiert wird, wird bei Augmented-Reality die Realitätswahrnehmung erweitert. Dies geschieht durch Einblendung zusätzlicher Informationen oder Überblenden von vorhandenen Objekten durch virtuelle Elemente. So steht dabei nicht die Schaffung einer möglichst perfekten virtuellen Welt im Vordergrund, sondern die Einblendung wichtiger Zusatzinformationen in das Sichtfeld des Benutzers. Diese Informationen werden dabei kontextabhängig eingeblendet, das bedeutet, die Einblendungen passen zum betrachteten Objekt, z.B. der Schaltplan der aktuell betrachteten Platine.

In Abbildung 2.1 ist die Einordnung der Augmented-Reality in das *Realitäts-Virtualitäts-Kontinuum* nach PAUL MILGRAM [MC99, S. 13] zu erkennen. Dieses Kontinuum reicht dabei von der realen Welt bis zur vollständigen virtuellen Welt. In dieses Kontinuum lassen sich Technologien einordnen, bei denen die reale mit der virtuellen Welt vermischt wird. Die gesamte Übergangsphase bezeichnet man als *Mixed Reality.* Je mehr man sich auf dieser Geraden von links nach rechts bewegt, desto mehr verdrängt die Virtualität die Realität. Die Augmented-Reality, wie sie in dieser Arbeit verwendet wird, befindet sich leicht rechts der realen Umgebung, auf der linken Seite des Graphen. Dabei wird die reale Welt als Ausgangsbasis genommen und nur mit virtuellen Objekten überlagert.

---

[1] reale Welt, die den bekannten physikalischen Gesetzen folgt
[2] Ausrichtung an einem einheitlichen Koordinatensystem

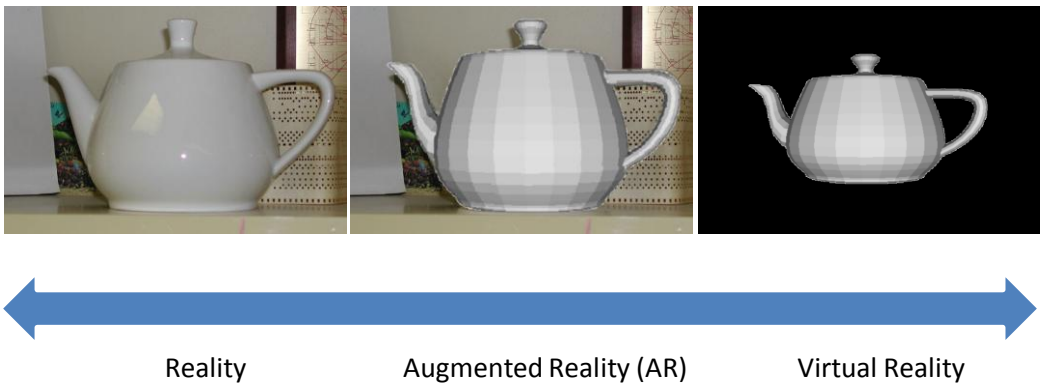

Abbildung 2.1 Einordnung von Augmented-Reality nach [MC99, S. 13]

Augmented-Reality Systeme bestehen aus drei Komponenten:

- Tracking-System (Abschnitt 2.2)
- Darstellungssystem (Abschnitt 2.3)
- Wissensbasis (Abschnitt 2.5)

Um mit den Objekten interagieren zu können, kommen in einigen Augmented-Reality Anwendungen zusätzliche Interaktionsgeräte zum Einsatz (Abschnitt 2.4).

2.2 Möglichkeiten des Trackings

Um die virtuellen Objekte kontextabhängig zu visualisieren, muss immer die exakte Position und Blickrichtung des Anwenders bekannt sein. Um dies zu gewährleisten, gibt es verschiedene Arten von Tracking-Systemen. Sie ermöglichen die deckungsgleiche Überlagerung von realem und virtuellem Bild. Zu diesem Zweck werden Sensoren benötigt, die sowohl die Lage als auch die Orientierung feststellen können (entsprechend Standpunkt und Blickrichtung des Betrachters). Sollen virtuelle und reale Objekte miteinander interagieren, so wird auch die Position und Orientierung dieser Objekte benötigt.

Dabei werden an die Sensoren von Tracking-Systemen verschiedene Anforderungen gestellt:

- hohe Aktualisierungsrate
- Je höher die Aktualisierungsrate desto flüssiger geschieht die Anpassung der virtuellen Welt an die reale Welt. Die Aktualisierungsrate wird dabei nicht nur durch den Sensor bestimmt, sondern auch durch die Übertragungsbandbreite vom Sensor zum Rechner.
- hohe Genauigkeit/Auflösung
- Je nach Anwendungsfall werden vom Tracking-System unterschiedlich genaue Auflösungen verlangt. Die Genauigkeit wird durch die Größe des Positions- und Orientierungsfehlers des Tracking-Systems bestimmt.
- große Reichweite
- Freiheitsgrade („Degrees of Freedom“, kurz DOF)
- Freiheitsgrade bezeichnen die Anzahl der Daten, die mit einem Sensor bestimmt werden können. Typischerweise besitzt ein Trackingsensor 6 Freiheitsgrade. Dazu zählen die Position sowie die Orientierung im 3D-Raum (siehe Abbildung 2.2).

- hohe Zuverlässigkeit
- geringe Latenz

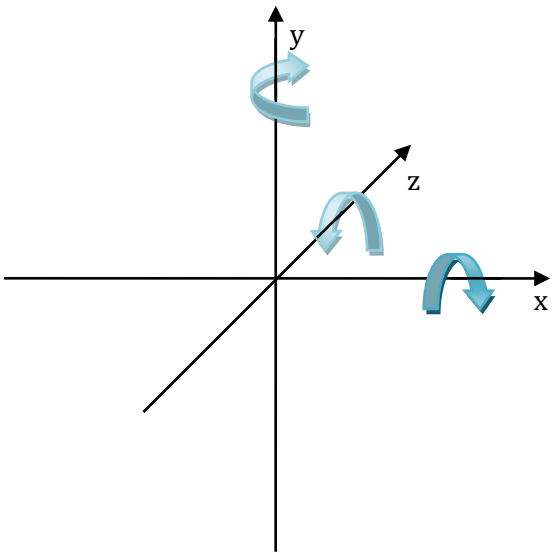

Abbildung 2.2 Die sechs Freiheitsgrade

Beim mobilen Einsatz von Augmented-Reality Systemen werden an ein Tracking-System noch weitere Anforderungen gestellt:

- geringe Größe, Formfaktor
- geringer Stromverbrauch
- Anforderungen an Infrastruktur
  Im mobilen Einsatz sollten die Tracking-Systeme keine oder nur geringe Anforderungen an eine vorgegebene Infrastruktur haben.
- keine Beeinflussung durch Umwelteinflüsse
  Tracking-Systeme im mobilen Einsatz sollten nicht durch Umwelteinflüsse beeinflusst werden. So sollte bei einem optischen Tracking-System die Erkennung nicht durch landschaftliche Gegebenheiten beeinflusst werden.

Im Folgenden werden die gebräuchlichsten Sensortypen aus dem Augmented-Reality Bereich vorgestellt. Die Gliederung und Sortierung der Sensoren richtet sich nach [RBG01].

2.2.1 Zeit- und Frequenzmessung

Optische oder akustische Signale werden von einem Emitter ausgesendet und von einem Transmitter empfangen. Durch die Messung der Signallaufzeiten lässt sich die Entfernung zwischen Emitter und Transmitter ermitteln. Ein Beispiel hierfür ist die Ultraschallmessung. Dabei wird ein Ultraschallsignal vom Sensor ausgesendet, von den Objekten im Raum reflektiert und vom Sensor wieder aufgenommen. Ein weiteres Beispiel ist die weitverbreitete Positionsbestimmung per GPS, dem Global Positioning System. Hier senden 24 Satelliten in gepulsten Abständen Signale aus, die Informationen tragen, wann sie abgeschickt wurden. Am Boden kann der Empfänger aus den verschiedenen empfangenen Signalen Unterschiede in der Laufzeit

ermitteln und durch Triangulation seine Position bestimmen. In Abbildung 2.3 ist ein GPS Modul zur Positionsbestimmung zu sehen.

Die Vorteile der Zeit- und Frequenzmessung sind vor allem eine hohe Aktualisierungsrate und eine große Genauigkeit. Ein Nachteil dieser Systeme ist die Starke Anfälligkeit gegenüber Störsignalen.

Abbildung 2.3 GPS Modul CW25-NAV der Firma Chronos Technology Ltd. [Chr07]

2.2.2 Mechanische Kopplungen

Bei der mechanischen Kopplung werden die zu trackenden Objekte über Stangen und Gelenke oder Seile mit festen Referenzpunkten verbunden. Über Beugungssensoren oder Spannungssensoren kann die Position des Objekts errechnet werden. In Abbildung 2.4 ist ein solches System zu sehen. Hierbei handelt es sich um ein PHANTOM® Omni™ Haptic Device, das es dem Benutzer erlaubt Virtuelle Objekte zu berühren und zu bearbeiten. Dies wird durch ein Force-Feedback (dt.: Kraft-Rückmeldung) System realisiert.

Vorteile dieser Systeme sind:

- hohe Präzision
- einfache technische und billige Realisierung
- hohe Aktualisierungsrate

Mechanische Kopplungen haben den Nachteil, da sie aufgrund der physischen Verbindung in ihrer Reichweite eingeschränkt sind.

Abbildung 2.4 PHANTOM® Omni™ Haptic Device der Firma SensAble Technologies, Inc. [Sen07]

2.2.3 Inertialsysteme

Inertialsysteme berechnen auf Grund ihres Ausgangszustandes Veränderungen relativ zu ihrer vorherigen Position oder Orientierung. Sie können jeweils nur Positionsänderungen, also relative Positionen, und keine absoluten Positionen messen. Dies ist ein großer Nachteil dieser Systeme, da sich Fehler in den Messdaten schnell fortpflanzen.

Beispiele für Inertialsysteme sind mechanische Gyroskope (Mechanische Kreisel) und Beschleunigungssensoren. In Abbildung 2.5 ist ein solcher Sensor zu sehen.

Inertialsysteme haben folgende Vorteile:

- keine zusätzlichen Geräte (z.B. Sender oder Kameras) benötigt.
- keine Beeinflussung durch äußere Einflüsse, wie z.B. Magnetfelder oder schlechte Lichtverhältnisse
- sehr kleine Messgeräte

Die Nachteile von Inertialsystemen sind:

- regelmäßige Rekalibrierung
- verwendete Technologie ist noch nicht ausgereift

Abbildung 2.5 Inertiales Messsystem der Consulting Measurement Technology GmbH [CMT07a]

2.2.4 Räumliche Scans

Unter den Begriff „räumliche Scans“ fallen viele Technologien, von der optischen Erfassung mit Hilfe einer Videokamera oder Infrarotkamera bis zur Abtastung mit Laserstrahlen. Bei der Videometrie, der Beobachtung eines Objektes mit einer Kamera, werden die relevanten Daten durch Bilderkennungsalgorithmen gewonnen. Um diesen Aufwand zu minimieren, benutzen viele Systeme Marker, die relevante Stellen am Objekt kennzeichnen.

In Abbildung 2.6 kann man einen Marker sehen, der bei solchen Systemen eingesetzt wird. Hierbei handelt es sich um einen Marker, wie er z.B. beim ARToolKit (siehe Abschnitt 4.1.1) zur Verwendung kommt. Wie das optische Tracking bei diesem System funktioniert wird im Abschnitt 2.8 näher beschrieben.

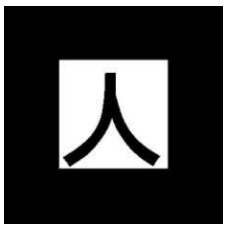

Abbildung 2.6 Beispiel für einen Marker wie er im ARToolKit vorkommt

Folgende Vorteile haben räumliche Scans:

- sehr präzise
- theoretisch hohe Aktualisierungsrate (eingeschränkt durch die Verarbeitung der großen Datenmenge)

Nachteile der räumlichen Scans sind:

- Funktionseinschränkung bei Verdeckung
- hoher Rechenaufwand

2.2.5 Direkte Feldmessung

Bei der direkten Feldmessung werden sowohl natürliche als auch künstliche physikalische Felder zur Positionsbestimmung und Orientierungsbestimmung genutzt. Beispiele für direkte Feldmessungen sind Gravitationsfeldsensoren, Kompasse und Magnetfeldsensoren (siehe Abbildung 2.7).

Vorteile der direkten Feldmessung sind:

- sehr kostengünstig
- kleine und leichte Sensoren
- hohe Aktualisierungsrate

Zwei Nachteile dieser Systeme sind:

- anfällig für Störungen gegenüber Unregelmäßigkeiten der genutzten Felder oder deren Beeinflussung durch äußere Felder
- Messbereich ist auf den Bereich des vom Emitter erzeugten Feldes beschränkt

Abbildung 2.7 Digitales High-Speed- Magnetometer CMT Consulting Measurement Technology GmbH [CMT07b]

2.2.6 Hybrid-Systeme

Bei den Hybrid-Systemen werden verschiedene Sensortypen miteinander kombiniert. Das Ziel dieser Kombinationen ist, die hohen Tracking-Anforderungen von Augmented-Reality Anwendungen unter den jeweiligen Rahmenbedingungen zu erfüllen. Die Kombination verschiedener Systeme bietet eine Vielzahl von Vorteilen gegenüber dem Einsatz einzelner Systeme. So können Daten anderer Sensoren genutzt werden, um die notwendige Berechnung zur Auswertung der Daten eines anderen Sensors zu vereinfachen oder zu beschleunigen. Die Daten eines langsamen aber dafür genaueren Sensors können genutzt werden, um die Daten eines schnelleren Sensors zu korrigieren. Außerdem lassen sich durch die Kombination verschiedener Sensordaten fehlende Freiheitsgrade ergänzen. Beim Ausfall eines Sensors stehen die Daten eines anderen als Backup zur Verfügung. Ein Nachteil dieser Systeme ist der größere Aufwand bei der Auswertung der Sensordaten.

2.2.7 Zusammenfassung

In Tabelle 4.1 sind noch einmal die behandelten Trackingverfahren zur besseren Übersicht, mit ihrer Funktionsweise sowie ihren Vor- und Nachteilen zusammengefasst.

Tracking-System	Funktionsweise	Vorteile	Nachteile
Zeit- und Frequenz-messung (Ultraschallsignal, GPS)	• Ermittlung der Signallaufzeiten Via Sender und Empfänger	• hohe Aktuali-sierungsrate • große Genauigkeit	• starke Anfälligkeit gegenüber Störsignalen
Mechanische Kopplung	• Messung erfolgt über Beugungs-sensoren oder Spannungs-sensoren	• hohe Präzision • einfache tech-nische und billige Realisierung • hohe Aktuali-sierungsrate	• eingeschränkte Reichweite
Inertialsysteme (Beschleunigungs-messer)	• Berechnung relativ zu der vorherigen Position oder Orientierung	• keine zusätzlichen Geräte nötig • unabhängig von Raumgröße • keine Beeinflus-sung durch äußere Einflüsse • kleine Messgeräte	• Fehleraddition • müssen oft rekalibriert werden
Räumliche Scans	• optische Erfassung mit Hilfe einer Videokamera oder Infrarotkamera • Abtastung mit Laserstrahlen	• sehr präzise • hohe Aktuali-sierungsrate • unabhängig von Temperatur	• Funktionsein-schränkung bei Verdeckung
Direkte Feldmessung	•	• kostengünstig • kleine und leichte Sensoren • hohe Aktuali-sierungsrate	• anfällig für Störungen und Beeinflussung durch äußere Felder • Messbereich ist auf den Bereich des vom Emitter erzeugten Feldes beschränkt
Hybrid Systeme	Verbindet die Funktionen der anderen Verfahren, um die Nachteile zu minimieren.		

Tabelle 2.1 Übersicht von Trackingverfahren für Augmented-Reality

2.3 Darstellungssysteme

Um die virtuellen Objekte mit der realen Welt verschmelzen zu können, werden spezielle Systeme zur Darstellung der Augmented-Reality benötigt. Drei dieser Möglichkeiten werden im Folgenden dargestellt.

2.3.1 Head-Mounted-Displays

Die *Head-Mounted-Displays* (HMD) unterteilen sich in drei Kategorien: Die erste Kategorie bilden die *Video-See-Through-Head-Mounted-Display.* Sie bestehen aus zwei kleinen Monitoren, die vor den Augen des Benutzers angebracht sind, und einer kleinen Kamera, die ebenfalls an dieser Brille befestigt ist. Über die Monitore wird einerseits die reale Szene, aufgenommen über die Kamera, als auch die überlagerten virtuellen Informationen angezeigt. Der Anwender kann also nicht mehr direkt die reale Welt sehen. Diese nimmt er nur noch über das Augmented-Reality System wahr.

In Abbildung 2.8 sieht man ein Konzeptdiagramm eines *Video-See-Through-HMDs*. Hier kann der Weg des Bildes von der Kamera über den Bildmischer, wo das reale Bild mit dem generierten Bild überlagert wird, bis zu den Monitoren verfolgt werden.

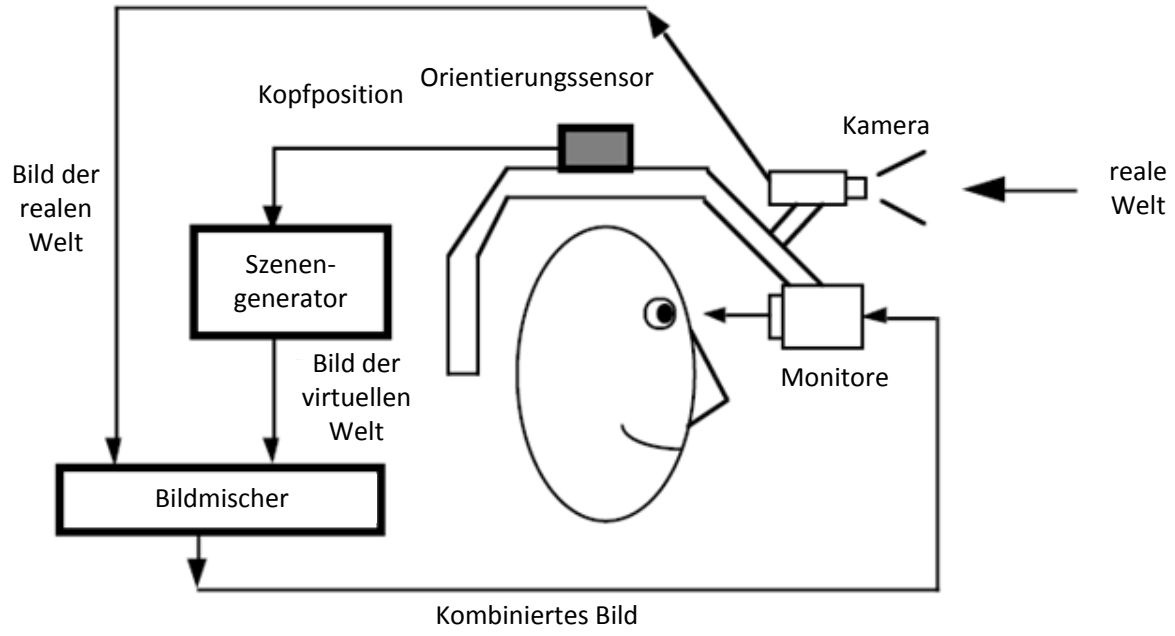

Abbildung 2.8 Diagramm eines Video-See-Through-Head-Mounted-Displays nach [Azu97]

Bei den *(Optical-)See-Through-Head-Mounted-Displays* handelt es sich um die zweite Kategorie der Head-Mounted-Displays. Bei diesen Systemen befindet sich anstelle der Monitore ein halb-durchlässiger Spiegel vor dem Auge des Benutzers. Dadurch kann der Benutzer die reale Szene direkt wahrnehmen. Über einen Monitor oberhalb des Spiegels wird das reale Bild mit den virtuellen Informationen überlagert. In Abbildung 2.9 ist dieses Prinzip vereinfacht dargestellt.

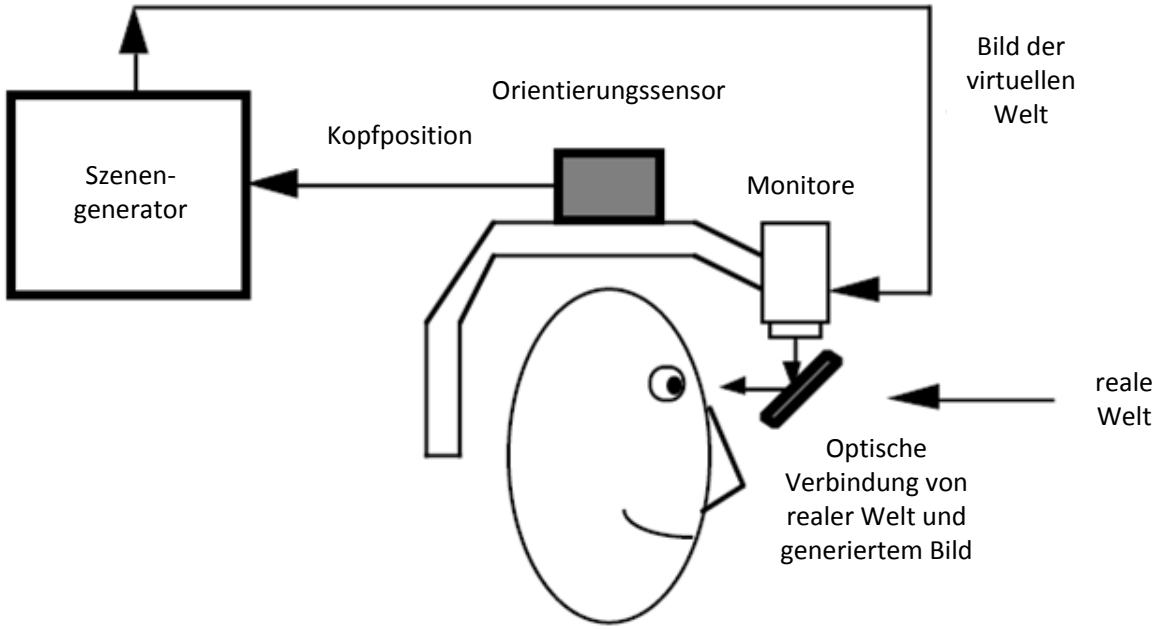

Abbildung 2.9 Diagramm eines Optical-See-Through-Head-Mounted-Displays [Azu97]

Die dritte Kategorie der HMDs bilden die sogenannten Virtual-Retinal-Displays (VRD). Im Gegensatz zu den beiden anderen Verfahren, wo das Bild auf einen Bildschirm projiziert wird und von dort erst ins menschliche Auge gelangt, wird bei einem VRD das Bild mit Hilfe eines Laserstrahls direkt auf die Netzhaut des Betrachters projiziert, wo es sich mit dem Bild der realen Welt vereint. In Abbildung 2.10 sind die einzelnen Komponenten eines VRD zu sehen.

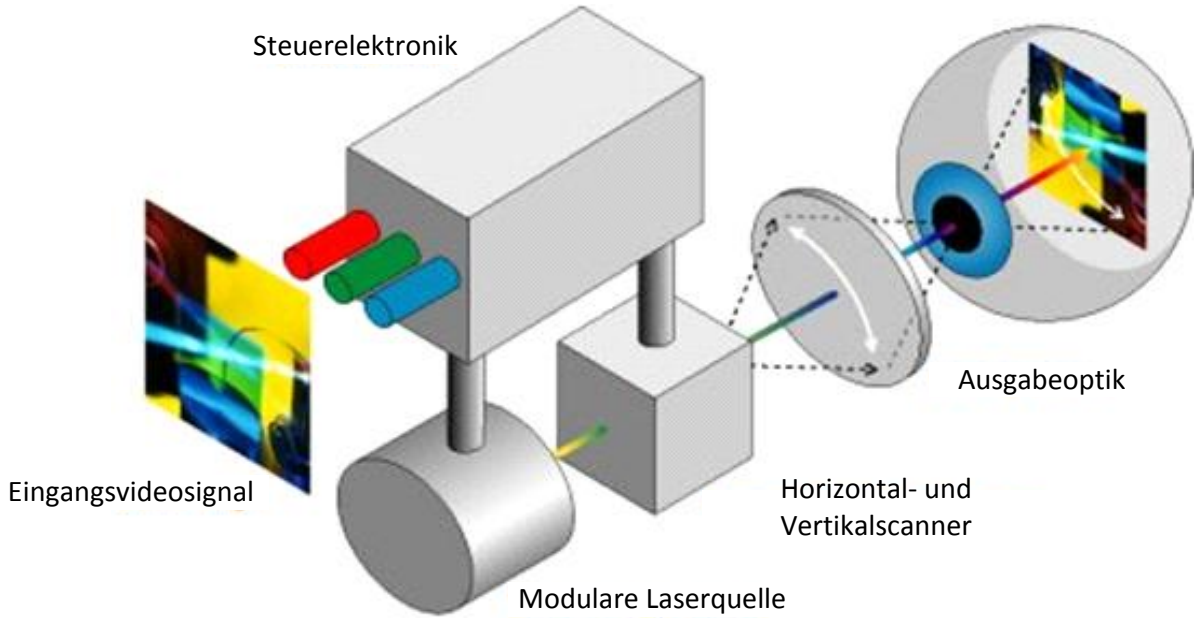

Abbildung 2.10 Grobes Schema eines Virtual Retinal Dispays [Fia07]

Die Steuerelektronik erhält das Eingangsvideosignal und extrahiert Informationen über die einzelnen Bildpunkte und deren Helligkeit. Diese Bildpunkte werden dann durch Helligkeits-

modulation des Laserlichts erzeugt. Im kommerziellen Bereich sind derzeit hauptsächlich monochrome VRDs üblich, bei denen lediglich ein roter Laserstrahl moduliert wird. Bei Farb-VRDs sind dies drei Laserstrahlen (rot, grün und blau) von denen jeder einzeln moduliert wird. Anschließend werden alle drei zu einem Strahl zusammengeführt.

Dieser Strahl wird dann auf die beiden Scanner umgeleitet. Der vertikale Scanner erzeugt dabei die Bildzeilen und der horizontale Scanner bewegt den Strahl schließlich in jeder Zeile von Pixel zu Pixel.

Das durch die Scanner erzeugte Bild wird anschließend durch eine Linse aufgefächert, da es sonst nur einen sehr kleinen Bereich der Netzhaut treffen würde. Über einen halbdurchlässigen Spiegel, der sich direkt vor dem Auge des Benutzers befindet, wird das Bild durch die Pupille direkt auf die Netzhaut projiziert. Wie auch bei den (Optical-)See-Through-Head-Mounted-Displays führt der halbdurchlässige Spiegel dazu, dass das Bild der realen Welt nur abgedunkelt, jedoch sonst in keiner Weise verändert wird und somit für den Benutzer weiterhin wahrnehmbar bleibt.

2.3.2 Projektorbasierte Systeme

Bei projektorbasierten Systemen werden die virtuellen Informationen mit Hilfe von Video- oder Laserprojektoren direkt auf die reale Szene projiziert. Das ist der entscheidende Vorteil dieser Systeme, da durch diese Art eine Vielzahl von Betrachtern die gleiche Augmented-Reality Umgebung wahrnehmen kann. Nachteil dieser Systeme sind die unterschiedlichen Blickwinkel der Betrachter. Dadurch können Objekte für verschiedene Betrachter an unterschiedlichen Stellen wahrgenommen werden. Außerdem kann die Szene durch einen anderen Betrachter verdeckt werden.

In Abbildung 2.11 erkennt man auf dem linken Bild die reale Szene. Man erkennt mehrere farblose Gegenstände. Auf dem rechten Bild ist dieselbe Szene, angereichert mit virtuellen Informationen, zu erkennen. Auf die reale Szene wurden verschiedene Schattierungen und Farben projiziert.

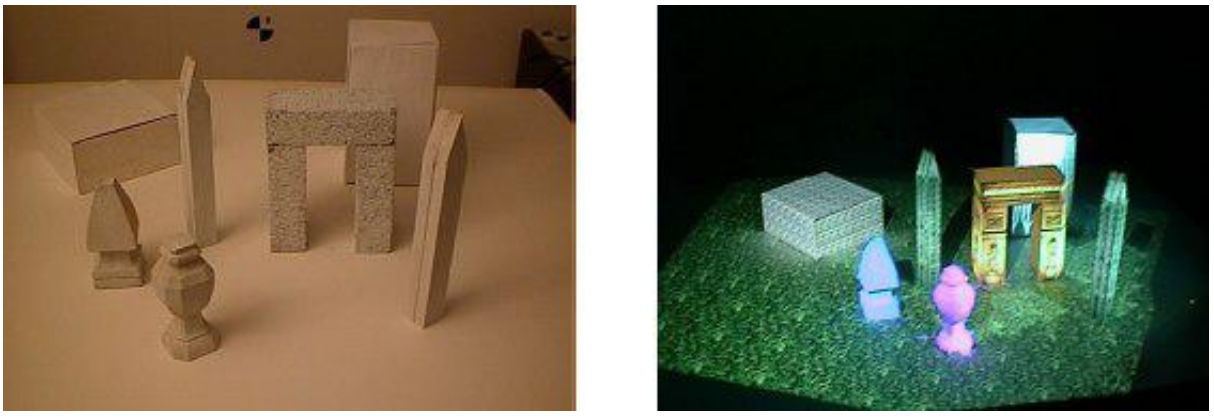

Abbildung 2.11 Projektorbasierte Augmented-Reality. Links erkennt man den Versuchsaufbau, rechts die Überlagerung durch verschiedene Schattierungen und Farben.

2.3.3 Handheld Systeme

Einige Augmented-Reality Systeme benutzen einen Handheld, Palm oder ein Smartphone mit einer angeschlossenen Kamera. Diese Systeme zielen auf mobile und leichte Augmented-Reality Anwendungen ab, die einen höheren Anspruch an Mobilität als der der Darstellung von hochkomplexen virtuellen Objekten haben.

So funktionieren sie wie die Video-See-Through-HMDs, da sie die reale Welt mit den überlagerten Informationen anzeigen. Durch die eingebaute Kamera nehmen sie die reale Umgebung auf und können dann die modifizierte Umgebung auf ihrem Display darstellen.

In Abbildung 2.12 sieht man die Verwendung eines PDAs als Darstellungssystem für eine Augmented-Reality Anwendung. Hierbei handelt es sich um ein mobiles, verteiltes Multi-User Augmented-Reality Spiel. Die Spieler kontrollieren virtuelle Züge, die sich auf realen hölzernen Miniatur-Gleisen bewegen. Diese Züge sind nur mit Hilfe der PDAs sichtbar. Die Projektseite ist unter [Tho07] zu finden.

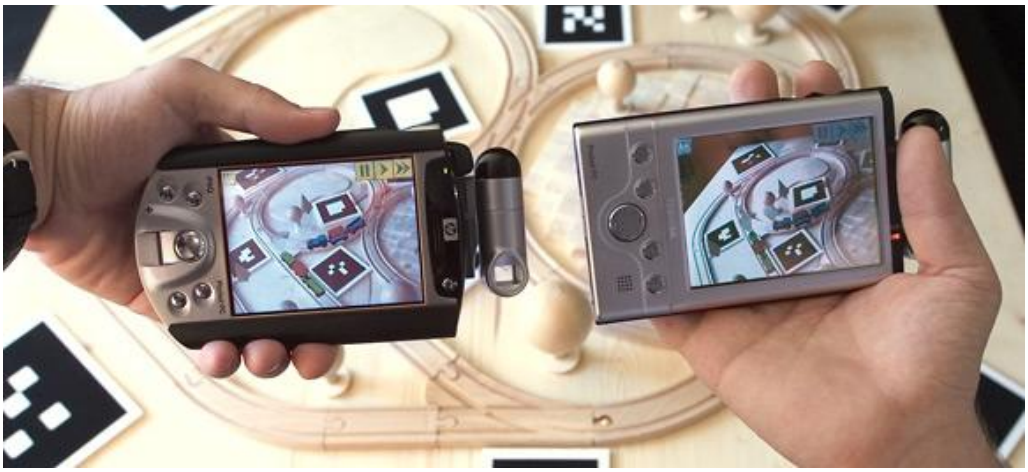

Abbildung 2.12 The Invisible Train [Tho07]

2.4 Interaktionsgeräte

Die meisten in der Augmented-Reality verwendeten Interaktionsgeräte wurden aus Virtual Reality-Systemen übernommen. Aus den technischen Gegebenheiten von Augmented-Reality Systemen und den Anforderungen von Augmented-Reality Anwendungen werden Interaktionsgeräte auch immer wieder neu entwickelt. Traditionelle Eingabegeräte aus dem Desktopbereich wie Joystick, Maus und Tastatur finden in der Augmented-Reality kaum Verwendung, da für Augmented-Reality Systeme Eingabegeräte mit 3D-Auflösung benötigt werden.

Im Folgenden wird ein Überblick über die in Augmented-Reality Anwendungen anzutreffenden Eingabegeräte gegeben.

2.4.1 Datenhandschuh

Zu den aus der virtuellen Realität bekannten Geräten gehören die verschiedenen Varianten der Datenhandschuhe, die auf vielfältige Art zur Systemsteuerung verwendet werden können. Ein

Datenhandschuh gestattet es dem Benutzer, virtuelle Objekte zu greifen oder auf andere Art zu manipulieren. Außerdem kann er zur Erkennung von Gesten benutzt werden.

Dazu wird die Haltung der Finger gemessen, um daraus Gesten abzulesen. Um die Stellung der Fingergelenke zu messen, gibt es verschiedene Verfahren. Eine Variante arbeitet mit Licht, das durch in den Handschuhfingern eingearbeitete Glasfaserschlaufen gesendet wird und je nach Fingerstellung die Lichtstärke verändert. Je stärker das Gelenk gekrümmt wird, desto weniger Licht tritt am Ende aus. In Abbildung 2.13 ist solch ein Datenhandschuh zur Erfassung von Hand- und Fingerbewegungen zu sehen.

Abbildung 2.13 CyberGlove® II Datenhandschuh der Firma Immersion Corporation [Imm07]

2.4.2 3D-Maus

Die Funktionsweise der 3D-Maus entspricht herkömmlichen 2D-Mäusen. Üblicherweise verfügt eine 3D-Maus über ein magnetisches Tracking-System zur Positions- und Lagebestimmung im Raum und eine oder mehrere Tasten zur Aktivierung von Funktionen, wie z.B. zum Selektieren von Objekten. Bei der in Abbildung 2.14 zu erkennenden Maus können durch Drücken, Ziehen, Kippen oder Drehen der Kappe drei verschiedene Freiheitsgrade angesprochen werden.

Abbildung 2.14 SpaceNavigator der Firma 3Dconnexion GmbH [Dco07]

2.4.3 Pen & Tablet

Der Nachteil des Datenhandschuhs und der 3D-Maus ist das Fehlen eines haptischen Feedbacks. Es können zwar Objekte manipuliert werden, aber es fehlt dabei das Gefühl, dieses Objekt auch wirklich greifen zu können.

Das Pen & Tablet - Eingabegerät besteht aus zwei unabhängig voneinander gehaltenen Teilen eines Interaktionsgerätes, bei dem mit einem Stift auf ein Tablett projizierte Interaktions-komponenten gesteuert werden. Der Vorteil gegenüber einer 3D-Maus oder einem Datenhandschuh ist, dass mit dem Stift das haptisch wahrnehmbare Tablett berührt wird. Der

Benutzer hat so nicht das Gefühl, mit dem leeren Raum zu interagieren. Das Tablett eignet sich sehr gut für Systemsteuerungsaufgaben es ähnelt einer Schalttafel, wie sie in Abbildung 2.15 zu erkennen ist, bei der die Bedienelemente virtuelle Objekte sind.

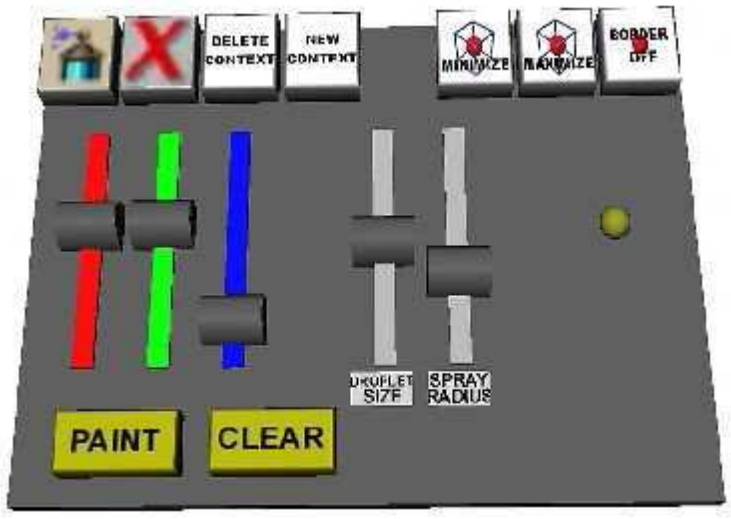

Abbildung 2.15 Anzeige von Interaktionselementen auf einem Tablet [Ulb02]

2.5 Inhaltsverwaltung

Für viele Anwendungsbereiche existieren bereits standardisierte Systeme, um große Informationsmengen effektiv erstellen, pflegen und diese an die Benutzer verteilen zu können. So gibt es für Web Publikationen bereits eine Vielzahl an Content Management Systemen. Bei Augmented-Reality Anwendungen sind die Inhalte oder Informationen meist noch fester Bestandteil der Software. Hier sind eine Standardisierung von Augmented-Reality Inhaltsbeschreibungen sowie die Entwicklung geeigneter Content-Management-Werkzeuge erforderlich [Pae06].

Viele Augmented-Reality Anwendungen könnten dabei auf schon vorhandene Datenbestände zugreifen. So sieht man in Abbildung 2.16 eine Video-Ansicht der Stadt Hannover. Diese Ansicht wurde mit Gebäudebeschreibungen aus einem Geographischen Informationssystems (GIS) erstellt.

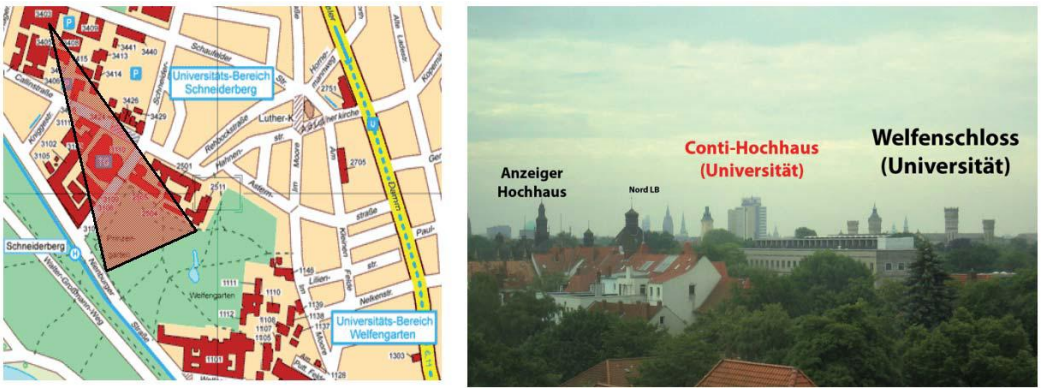

Abbildung 2.16 Sicht auf die Stadt Hannover mit angereicherten Daten aus einem GIS [Pae06]

2.6 Die verschiedenen Koordinatensysteme

In einer Augmented-Reality Umgebung müssen verschiedene Koordinatensysteme (siehe Tabelle 2.2) kombiniert werden. Im Folgenden werden die Koordinatentransformationen anhand eines Marker basierten Tracking-Systems erläutert.

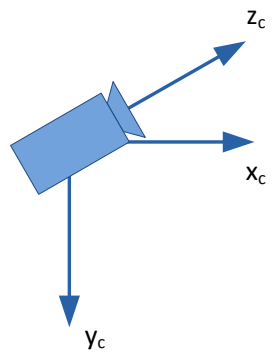

Abbildung 2.17 Kamera-Koordinatensystem

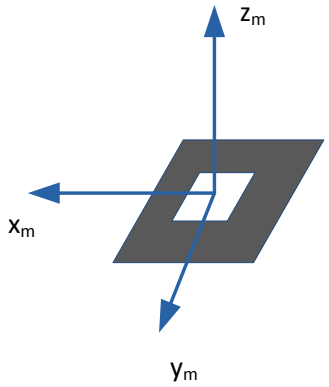

Abbildung 2.18 Marker-Koordinatensystem

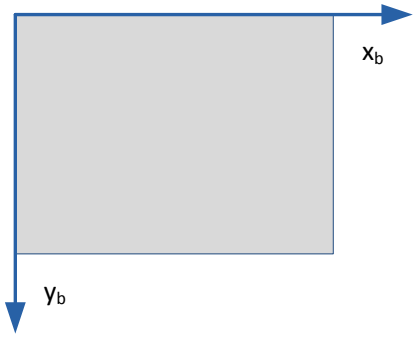

Abbildung 2.19 Bildschirm-Koordinatensystem

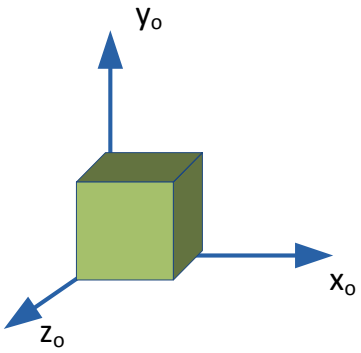

Abbildung 2.20 Objekt-Koordinatensystem

Tabelle 2.2 In einem AR-System vorkommende Koordinatensysteme

Um die Position und Ausrichtung eines Markers mit Bezug auf das Kamera-Koordinatensystem zu erhalten, werden die Eckpunkte der erkannten Marker im Kamera-Koordinatensystem abgebildet. Sind alle vier Eckpunkte eines Markers im abgebildet, kann anhand der bekannten Größe des Markers sowie den Kameraparametern, die Entfernung des Markers zur Kamera bestimmt werden. Somit erhält man eine Transformationsmatrix, welche die Position und Ausrichtung des Markers im Kamera-Koordinatensystem definiert. Abbildung 2.21 veranschaulicht den Vorgang der Positionsbestimmung eines Marker-Eckpunktes im Kamera-Koordinatensystem.

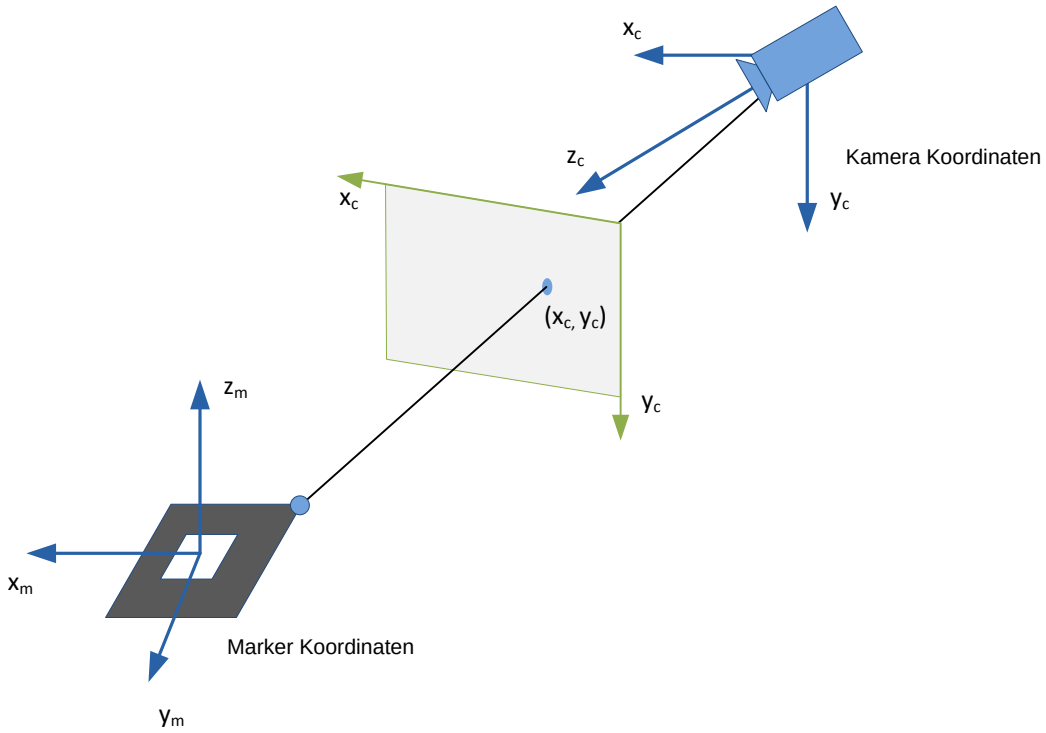

Abbildung 2.21 Beziehung zwischen Marker und Kamera Koordinatensystem

Um die Position der Kamera im Marker-Koordinatensystem zu bestimmen, muss nur die Inverse der ermittelten Transformationsmatrix des Markers bestimmt werden.

Für die Darstellung von Objekten in der virtuellen Welt, gibt es auch zwei Koordinatensysteme die in Beziehung gesetzt werden müssen. Dabei handelt es sich ebenfalls um ein Kamera-Koordinatensystem sowie um ein Objekt-Koordinatensystem. Hier ist die Kamera aber keine physische, sondern eine virtuelle Kamera. Diese nimmt die virtuelle Szene auf. Das Objekt-Koordinatensystem ist mit dem des Markers zu vergleichen. Die Koordinatensysteme beider Kameras sind deckungsgleich, da die virtuelle Szene die Aufnahme der realen Kamera überlagert. Wird also die Transformationsmatrix eines Markers auf ein virtuelles Objekt angewandt, so wird dieses an derselben Position wie der Marker angezeigt. Eine eventuelle Abweichung muss durch eine Messung und Kalibrierung behoben werden. Solche Abweichungen können z.B. durch geänderte oder falsche Kameraparameter auftreten.

Um das erzeugte Bild der virtuellen Kamera auf dem Bildschirm darstellen zu können, müssen wiederrum zwei Koordinatensysteme in Beziehung gesetzt werden. Hierbei handelt es sich um das 3D-Koordinatensystem der virtuellen Kamera und um das 2D-Koordinatensystem des Bildschirms. Um die virtuelle 3D-Szene auf dem 2D-Bildschirm ausgeben zu können, werden die Punkte des 3D-Szene auf Punkte einer gegebenen Ebene mit Hilfe der Projektion abgebildet.

In Abbildung 2.22 ist noch einmal der gesamte Vorgang vom Marker-Koordinatensystem bis zum Bildschirm-Koordinatensystem dargestellt.

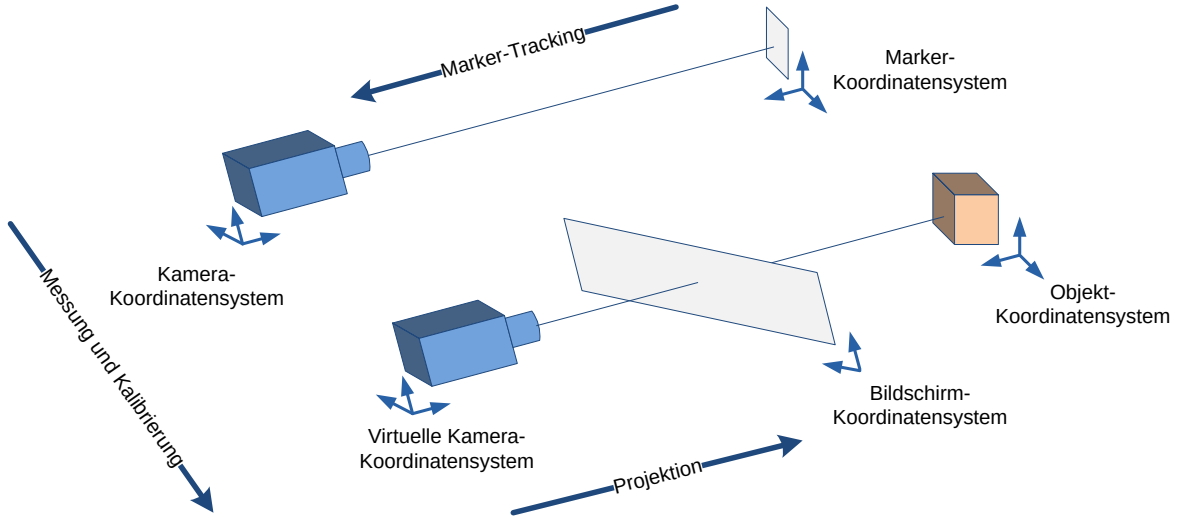

Abbildung 2.22 Verschiedene Koordinatensysteme und ihre Transformationen

2.7 Struktur einer Augmented-Reality Anwendung

In Abbildung 2.23 ist der vereinfachte Ablauf in einer Augmented-Reality Anwendung dargestellt. Ausgangsbasis einer jeden Augmented-Reality Anwendung stellt ein Bild der Realität dar. Mit Hilfe von Sensoren erfolgt das Tracking oder auch die Lagebestimmung. Im Abschnitt 2.8 wird das Tracking am Beispiel eines optischen Tracking-Systems näher beschrieben. Dabei kann es sich um die Position- oder Orientierungsbestimmung eines Benutzers, der Kamera oder eines beliebigen anderen Objektes handeln. Das hängt jeweils von dem Einsatz der Anwendung ab. Die beim Tracking gewonnenen Lage- und Orientierungsdaten werden anschließend in die virtuelle Umgebung übernommen. Somit kann die reale und die virtuelle Welt an einem einheitlichen Koordinatensystem ausgerichtet und kombiniert werden. Dies wird auch als Registrierung bezeichnet. Im letzten Schritt wird das Ergebnis gerendert und dem Benutzer ausgegeben.

Bei einem Optical-See-Through System wird das Bild der realen Welt nicht mit ausgegeben, da dieses direkt gesehen wird. Es erfolgt also nur eine Ausgabe der virtuellen Welt. Die Kombination beider Welten erfolgt dann erst im Auge des Benutzers.

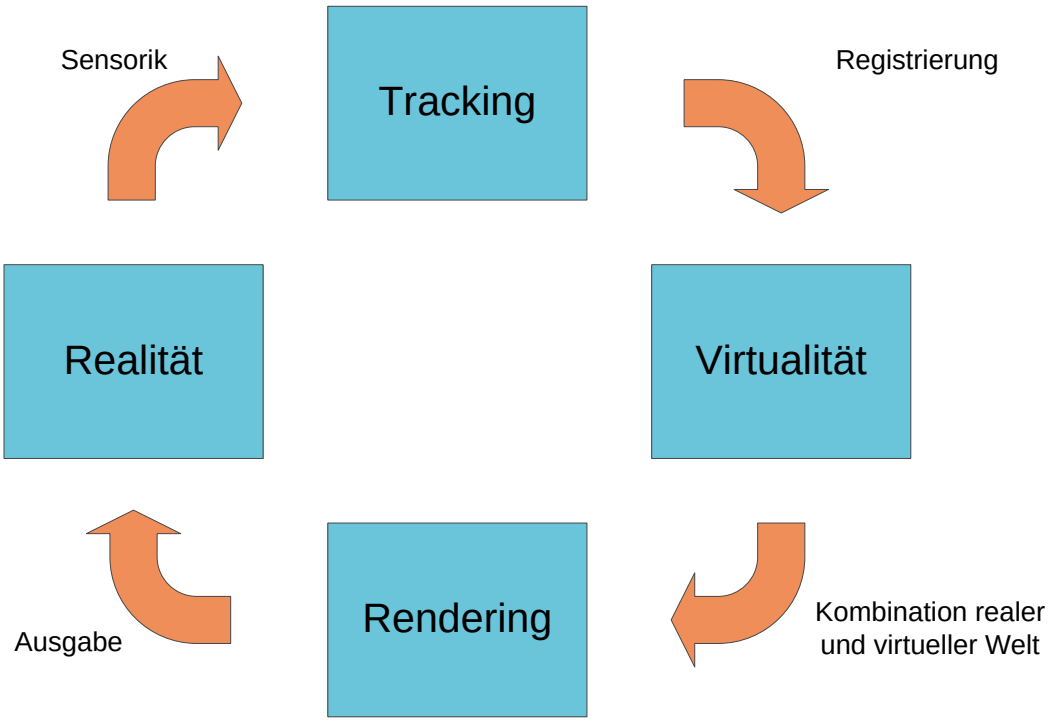

Abbildung 2.23 Ablaufschema eines typischen Augmented-Reality Systems

2.8 Beschreibung des Trackingvorganges am Beispiel des ARToolKits

In Abschnitt 4.1 werden verschiedene markerbasierte optische Tracking-Systeme vorgestellt. Da in dieser Arbeit das ARToolKit zum Einsatz kommt, soll die Funktionsweise eines marker-basierten optischen Tracking-Systems an diesem erklärt werden.

2.8.1 Vorraussetzungen

Für den Einsatz des ARToolKits wird lediglich eine Videokamera bzw. Webcam benötigt. Die Marker können mit Hilfe eines Bildbearbeitungsprogrammes und eines Druckers selbst hergestellt werden. Bei den Markern handelt es sich um ein eindeutiges Muster, welches innerhalb eines schwarzen Rahmens abgebildet ist, wie in Abbildung 2.6 zu sehen. Um die einzelnen Marker unterscheiden zu können, müssen der Software die Muster bekannt sein. Daher gibt es für jeden Marker eine Datei, welche ein Bild des Musters enthält. Somit kann die Software durch einen Vergleich den entsprechenden Marker erkennen.

2.8.2 Erkennen eines normalen Markers

In Abbildung 2.23 ist der genaue Ablauf der Erkennung sowie der Bestimmung der Transformationsmatrix aller Marker die auf einem Videobild zu sehen sind dargestellt. Im Folgenden wird der Vorgang der Markererkennung vereinfacht erläutert.

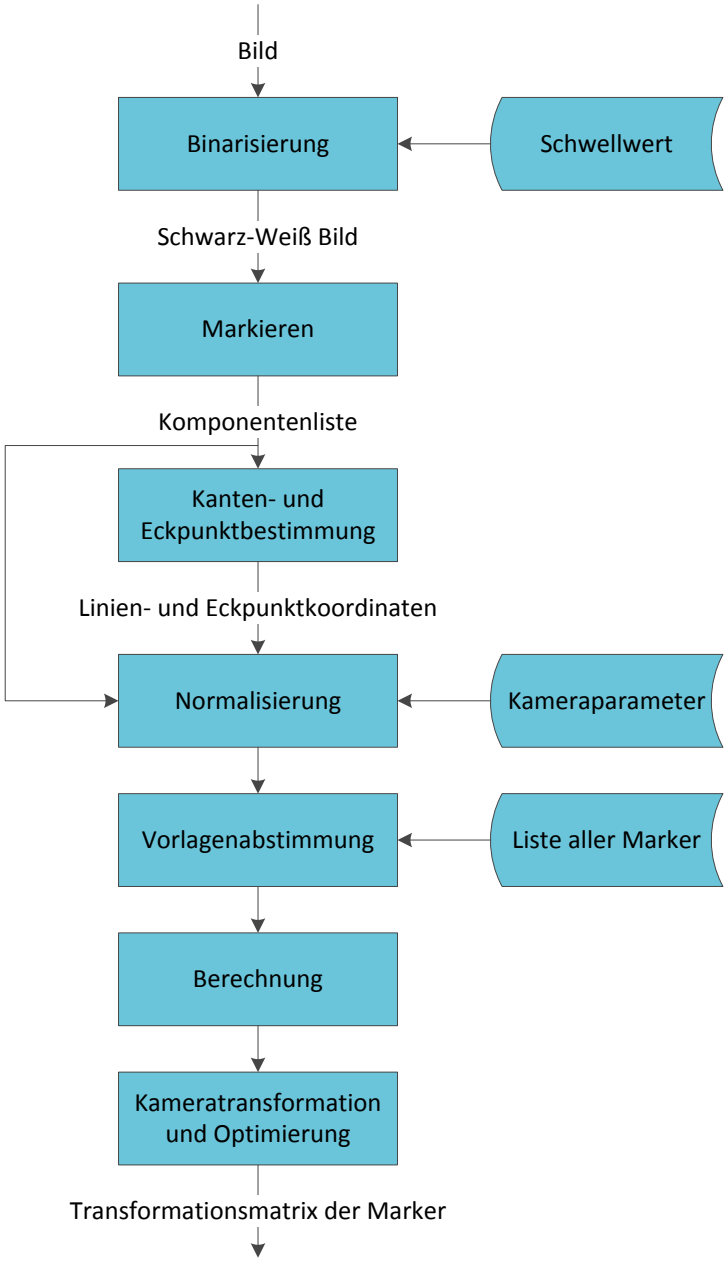

Abbildung 2.24 Funktionsweise des ARToolKits

Für das Matching der Marker wird das Videobild (siehe Abbildung 2.25 – Links) in ein Binärbild mit nur zwei Helligkeitsstufen (Schwarz und Weiß) umgewandelt. (siehe Abbildung 2.25 – Mitte). Dies geschieht, indem mit Hilfe eines Schwellwertes alle Pixel, die heller als der Schwellwert sind, auf Weiß und alle dunkleren Pixel auf Schwarz gesetzt werden. Auf diesem Bild wird dann nach viereckigen Flächen gesucht (siehe Abbildung 2.25 – Mitte). Dabei muss es sich aber nicht immer um einen relevanten Marker handeln.

Im Anschluss wird das Muster innerhalb dieser Fläche mit den Mustern in den gespeicherten Dateien verglichen. Wird eine Übereinstimmung gefunden, so werden Translation und Rotation abhängig zum Kamerakoordinatensystem bestimmt und gespeichert. Mit diesen Daten kann dann z.B. der gefundene Marker mit einem virtuellen Objekt überlagert werden (siehe Abbildung 2.25 - Rechts). Sind mehrere Übereinstimmungen vorhanden, entscheidet der Grad

der Ähnlichkeit, welches Rechteck letztlich ausgewählt wird. Das heißt, kommt ein Marker mehrmals im Sichtbereich der Kamera vor, so wird er trotzdem nur einmal erkannt. Eine zufällig erkannte Form wird also an dieser Stelle meistens aussortiert, da ihre Ähnlichkeit zu den gespeicherten Mustern selten größer ist als die der richtig erkannten Marker. Dieser Prozess wiederholt sich für jeden Videoframe.

Abbildung 2.25 (a) das Bild der Videokamera, (b) Videobild nach Schwellwertanwendung (Schwellwert: 100) mit einem gefundenen Marker (rot markierte Fläche), (c) der gefundene Marker wird mit einem einfachen 3D-Objekt überlagert.

2.8.3 Erkennen eines Multimarkers

Desweiteren ist es beim ARToolKit möglich, mehrere Marker zu einem großen „Multimarker“ zusammenzufügen. In Abbildung 2.26 ist ein solcher Multimarker zu sehen. Er besteht aus sechs einfachen Markern. Der Vorteil dieser Multimarker ist, dass nicht mehr der komplette Marker erkannt werden muss. Es reicht, wenn mindestens ein Marker des Multimarkers erkannt wird. In diesem Fall ist die genaue Lage des gesamten Markers bestimmbar. Er kann wie ein ganz normaler Marker behandelt werden. Wird er erkannt, so wird, wie bei einem normalen Marker, seine Position und Lage bestimmt.

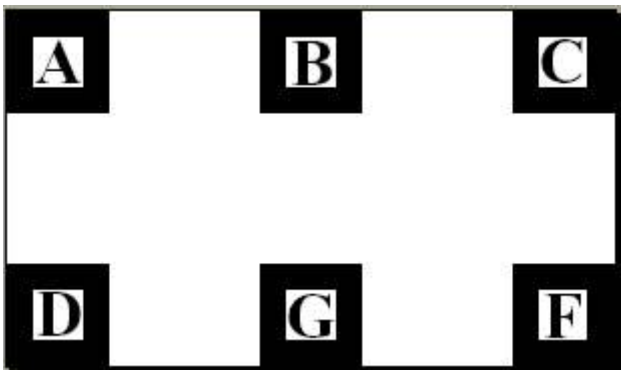

Abbildung 2.26 Multimarker bestehend aus sechs einzelnen Markern

Außerdem ist es möglich die Position und Lage jedes Teilmarkers zu bestimmen. Dies ist möglich, da die genaue Lage der einzelnen Marker auf dem Multimarker im Vorfeld durch eine Konfigurationsdatei festgelegt ist. In Abbildung 2.27 ist solch eine Konfiguration dargestellt.

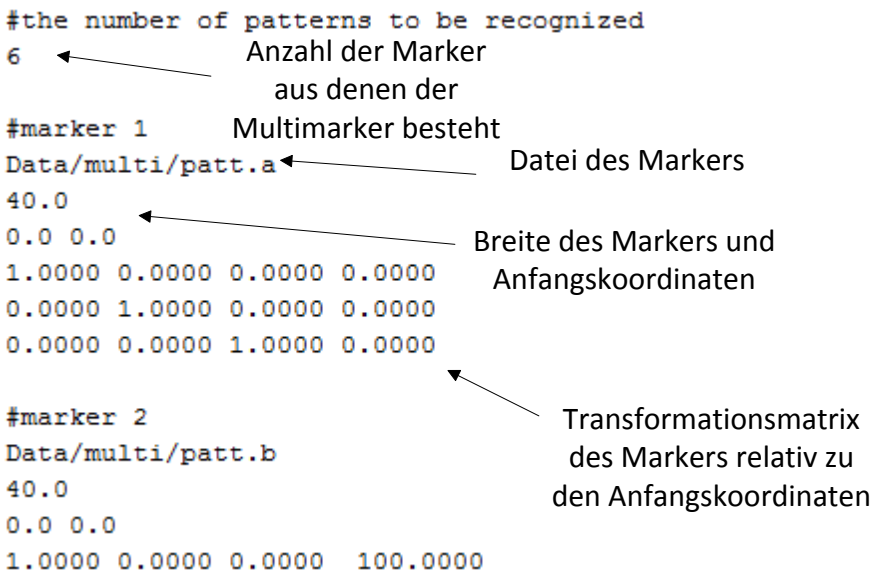

Abbildung 2.27 Multimarker Konfigurationsdatei

2.8.4 Grenzen des Trackingverfahrens

Das fehlerfreie Tracking des Markers funktioniert aber nur, solange der Marker für die Kamera sichtbar ist. Wird der Marker teilweise oder ganz verdeckt, z.B. durch einen Gegenstand, der sich zwischen dem Marker und der Kamera befindet, kann der Marker nicht mehr richtig erkannt werden. Dies kann auch schon dann eintreten wenn nicht das Muster innerhalb des Markers, sondern nur der Rahmen verdeckt wird.

Eine weitere Beschränkung dieses Trackingverfahrens ist die Abhängigkeit zwischen der Größe eines Markers und seiner Entfernung zur Kamera. Je größer ein Marker ist, desto weiter entfernt kann er von der Kamera sein, um noch erkannt zu werden. Tabelle 2.3 veranschaulicht den Zusammenhang zwischen der Größe eines Markers und der maximalen Entfernung zwischen ihm und der Kamera für ein korrektes Tracking.

Größe des Markers (in cm)	Reichweite (in cm)
6,99	40,64
8,89	63,5
10,8	86,36
18,72	127

Tabelle 2.3 Zusammenhang zwischen Markergröße und Reichweite beim ARToolKit nach [Lam07]

Die Reichweite wird ebenfalls durch die Komplexität der Marker beeinträchtigt. Je einfacher das Muster eines Markers ist, desto besser kann es auch noch in einer größeren Entfernung erkannt werden. Auch die Orientierung des Markers zur Kamera spielt bei der Erkennung eine große Rolle. Je flacher der Winkel zwischen Marker und Kamera ist, desto weniger Fläche des Markers ist sichtbar und desto schwerer wird das korrekte Tracking des Markers.

Kapitel 3 Augmented-Reality in Industrie, Forschung und Medizin

Die Anfänge von Augmented-Reality reichen auf die Simulatoren zur Ausbildung von Kampffliegern und auf die Computergrafik-Forschungen der 60er Jahre zurück. Der Wissenschaftler Ivan Sutherland veröffentlichte 1965 die theoretischen Grundlagen unter dem Titel „The Ultimate Display“ [Sut65]. Seit Sutherlands Veröffentlichung und seiner Erfindung des ersten am Kopf getragenen Displays [Sut68], wurden weitere Anstrengungen unternommen, die Rechnerleistung von einem externen Computer in eine Maschine am Körper zu verlagern. Dies sollte dem User ermöglichen, durch die virtuelle Welt genauso zu navigieren wie durch die reale Welt, nämlich mit allen Sinneseindrücken: bewegen (gehen), sehen, hören, berühren, verwenden (von Gegenständen), usw.

Seit 1992 Boeing den Begriff Augmented-Reality prägte [Miz00], ist dieses Thema ein großer Forschungsschwerpunkt für viele Institutionen. Da es den Rahmen dieser Arbeit sprengen würde, alle Bereiche auch nur ansatzweise anzuschneiden, wird im folgenden ein kurzer Überblick über die Einsatzmöglichkeiten von Augmented-Reality im industriellen Umfeld (Abschnitt 3.1), in der Medizin (Abschnitt 3.2) und in anderen Bereichen (Abschnitt 3.3) gegeben. Anschließend werden noch zwei wegweisende Projekte aus der Industrie (Abschnitte 3.4.1 und 3.4.2) sowie zwei medizinische Projekte (Abschnitte 3.4.3 und 3.4.4) kurz vorgestellt.

3.1 Augmented-Reality im industriellen Umfeld

Für die Aufgabe der geeigneten Informationsdarstellung ist Augmented-Reality eine passende Lösung. Sie kann in praktisch allen Bereichen, in denen es um die Anreicherung des normalen Umfeldes mit zusätzlichen kontextbezogenen Informationen geht, eingesetzt werden. In den Cockpits der US-amerikanischen Militärjets wurde Augmented-Reality zum ersten Mal eingesetzt. So wurden den Piloten wichtige Flugdaten sowie das Zielerfassungssystem direkt in die Frontscheibe eingeblendet.

Heutzutage wird Augmented-Reality aber nicht nur im militärischen Umfeld genutzt. Vor allem in der Industrie benötigt man eine immer höhere Qualifikation des Personals und der unterstützenden Technik. Grund hierfür sind die immer komplexer werdenden Arbeits- und Produktionsschritte. Dadurch fallen immer mehr Informationen, z.B. über die Montagereihenfolge, Teilelisten oder auch Schaltpläne an, die der jeweiligen beteiligten Person idealerweise immer zum korrekten Zeitpunkt und am richtigen Ort zur Verfügung gestellt werden müssen. Schon seit einigen Jahren hält daher Augmented-Reality immer mehr Einzug in der Industrie. Einsatzbereiche sind neben der Entwicklung und Produktion vor allem auch der Servicebereich.

Bei Virtual Reality Anwendungen müssen die haptischen Eindrücke durch spezielle Hardware realisiert werden (Mixed-Mock-Up[3]). Mit Hilfe von Augmented-Reality können solche Anwendungsprobleme einfacher gelöst werden, da die haptischen Eindrücke durch reale Objekte abgedeckt werden. Außerdem ist der direkte Vergleich zwischen Versuchsergebnissen und vorher berechneten Resultaten möglich. So lässt sich bei einem Crash-Test sofort erkennen, ob Differenzen zwischen den Verformungen am realen Fahrzeug und der vorher berechneten Verformung, bestehen.

Genauso kann im Produktionsprozess immer der nächste durchzuführende Arbeitsschritt angezeigt werden bzw. der aktuelle Produktionsschritt kontrolliert und Fehler direkt an dem betreffenden Objekt dargestellt werden. So kann die Trainingsphase eines Monteurs erheblich verkürzt und vereinfacht werden.

Im Servicebereich bietet der Einsatz von Augmented-Reality wie auch bei der Produktion die Möglichkeit einem Techniker durch die Überlagerung realer Objekte die nächsten Service- und Wartungsschritte zu erläutern. So sieht man in Abbildung 3.1, wie Augmented-Reality einen Monteur unterstützen kann. Die angereicherten Bilder werden über das HMD dem Monteur angezeigt. Dabei wird das nächste auszubauende Bauteil markiert (Abbildung in der Mitte) und der Ausbaupfad angezeigt. Dadurch kann der Monteur, unterstützt durch die Demontage-informationen, das Bauteil ausbauen.

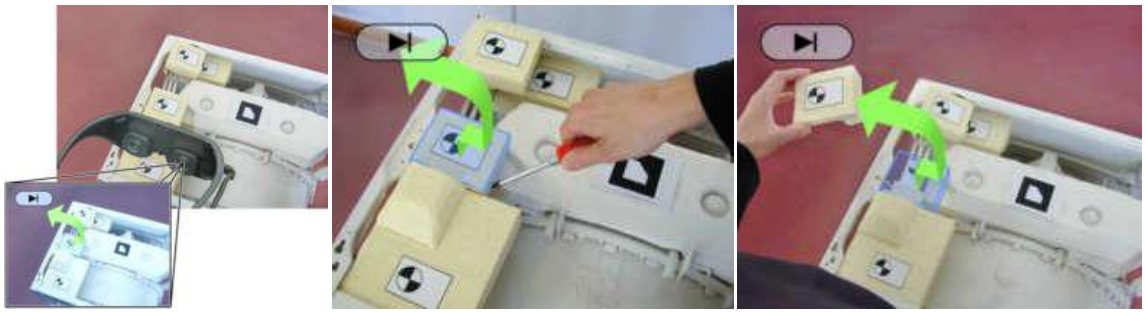

Abbildung 3.1 Beispiel für die Unterstützung eines Monteurs bei der Wartung einer Waschmaschine [Ruh04, S. 24]

3.2 Augmented-Reality in der Medizin

Die Medizin stellt für Augmented-Reality Systemen den wohl wichtigsten und auch zukunftsträchtigen Bereich dar. So kann ein Augmented-Reality System einem Mediziner bei der Planung und Vorbereitung aber auch bei der Durchführung einer Operation helfen.

3.2.1 Intraoperative Unterstützung

Der Chirurg kann während einer Operation durch ein Augmented-Reality System Unterstützung bei der Navigation und zur Orientierung bekommen. Ein Anwendungsbeispiel dafür ist die Minimal-invasive Chirurgie. Hierbei handelt es sich um interventionelle Eingriffe, bei denen durch minimale Zugänge für die Instrumente der Grad der Verletzungen gering gehalten wird.

[3] Verknüpfen realer und virtueller Elemente

Beispiele hierfür sind die Entnahme von Gewebeproben und Operationen im Brust- und Bauchraum. Da der Chirurg dabei keinen direkten Blick auf das Operationsgebiet hat, muss er die Operation über einen Monitor durchführen.

Mit Hilfe von Augmented-Reality können dem Chirurgen die Informationen über die exakte Position des Instruments, mit Hilfe von geeigneten Darstellungssystemen, direkt in sein Sichtfeld eingeblendet werden. Er sieht also praktisch „durch“ die Haut hindurch. Des weiteren können auch noch andere Informationen über den Patienten und die Operation eingeblendet werden. Dazu zählen Bilder, die mittels Ultraschall, Magnetresonanztomographie (MRT) oder Computertomographie (CT) gewonnen wurden. Dadurch können Operationen schneller und auch sicherer durchgeführt werden.

In Abbildung 3.2 sieht man auf dem linken Bild eine herkömmliche Operation über einen Monitor. Dabei muss der Arzt ständig auf den Monitor schauen, um die Operation durchführen zu können. Auf dem rechten Bild wird die Operation mit Hilfe eines Augmented-Reality Systems durchgeführt. Hier braucht der Arzt seinen Blick nicht abzuwenden, um zu sehen, an welcher Stelle sich seine Instrumente befinden. Diese Informationen werden direkt in sein Sichtfeld eingeblendet.

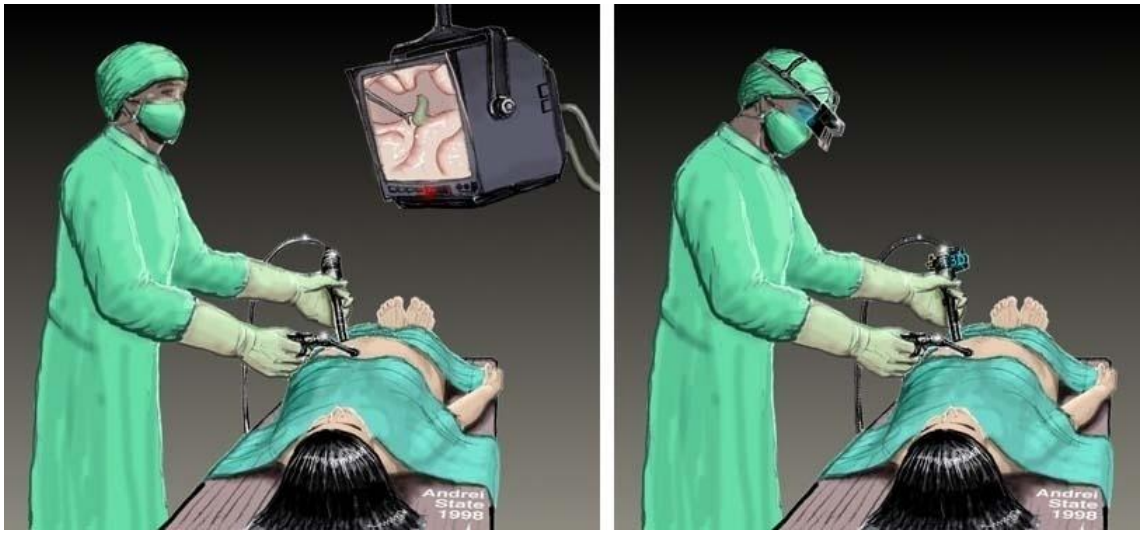

Abbildung 3.2 links: herkömmliche Operation, rechts Operation mit einem Augmented-Reality System [Fuc07].

3.2.2 Chirurgisches Training

Augmented-Reality kann den Ärzten beim Training und bei der Schulung helfen. Dabei arbeiten die Ärzte nicht an Leichen, wie es immer noch üblich ist, sondern an realitätsnahen Modellen mit allen diagnostisch relevanten anatomischen Strukturen (siehe Abbildung 3.3). Organe und Gewebe werden dabei als virtuelle Objekte eingeblendet. Der Arzt gewinnt damit für spätere Eingriffe Sicherheit und baut Hemmungen ab.

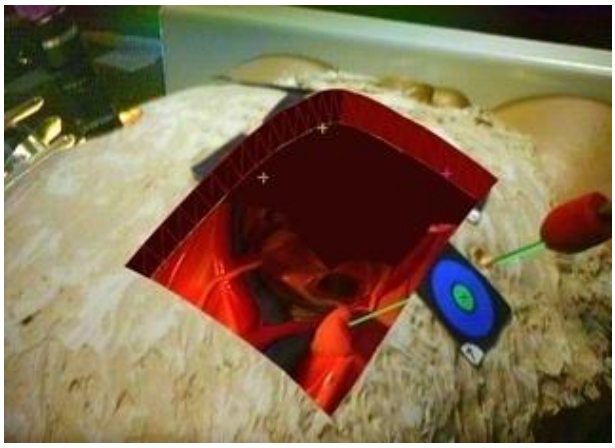

Abbildung 3.3 Biopsie mit Hilfe von Augmented-Reality an einem Versuchsaufbau [Fuc07]

3.2.3 Präoperative Planung

Der Chirurg kann nicht nur im Training und während einer Operation durch Augmented-Reality unterstützt werden, sondern auch schon bei der Operationsplanung. So kann vor einem Eingriff bei einem Patienten jeder einzelne operative Schritt an einem virtuellen Patienten simuliert werden. Durch die Simulierung immer wieder neuer Varianten des geplanten Eingriffes ist es dem Chirurgen möglich, den sichersten und effektivsten operativen Ansatz zu wählen.

3.3 Augmented-Reality in anderen Bereichen

Aber nicht nur in der Industrie, Forschung und der Medizin gibt es Einsatzmöglichkeiten für Augmented-Reality. Weitere Einsatzgebiete finden sich in allen Bereichen des täglichen Lebens. So könnte ein Tourist bei einem Stadtspaziergang das Aussehen der aktuellen Straße zu einer gewissen Zeit sehen oder er bekommt die Speisekarte des Restaurants eingeblendet, an dem er gerade vorbei läuft. Ein Computerspieler müsste nicht vor dem Computer sitzen, um Pacman zu spielen. Er könnte selber in die Rolle des Pacman schlüpfen (siehe Abbildung 3.4).

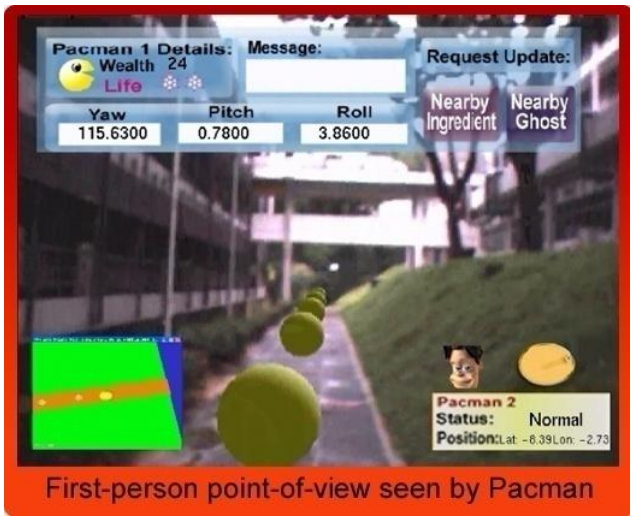

Abbildung 3.4 Human Pacman – Die Umgebung, durch die sich der Spieler bewegt, ist real, nur die Objekte und die überlagerten Informationen sind virtuell [Nat07].

Auch heute schon kommt Augmented-Reality in Computerspielen zum Einsatz, so wird z.B. eine virtuelle Landkarte oder Lageplan in das Sichtfeld des Spielers eingeblendet. Aber auch in der Archäologie (vgl. [Pap05], und [Bim02]), im Fernsehen (vgl. [Gra05]) und bei Navigationssystemen stößt man immer wieder auf Augmented-Reality. In Abbildung 3.5 sind noch einmal mögliche Anwendungsgebiete für Augmented-Reality aufgeführt.

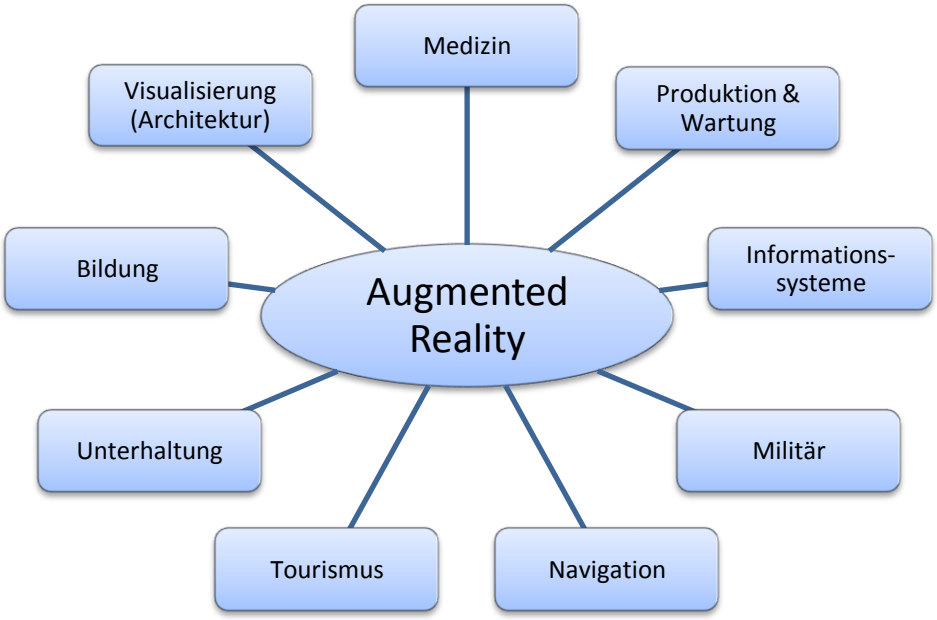

Abbildung 3.5 Anwendungsgebiete für Augmented-Reality

3.4 Beispiele für Augmented-Reality Anwendungen und Projekte

3.4.1 ARVIKA

Bei ARVIKA handelte es sich um ein vom Bundesministerium für Bildung und Forschung (BMBF) gefördertes Leitprojekt. Es sollten Augmented-Reality Technologien zur Unterstützung von Arbeitsprozessen in der Entwicklung, Produktion und dem Service für komplexe technische Produkte und Anlagen benutzerorientiert und anwendungsgetrieben erforscht und realisiert werden. An dem Projekt waren über 20 Unternehmen, unter anderem DaimlerChrysler, EADS/Airbus, Siemens, VW und Forschungseinrichtungen wie die TU München oder die RWTH Aachen beteiligt.

ARVIKA ist das Ergebnis der zweckmäßigen Zusammenführung von AUGRE und ASSIST, zweier Vorschläge für den BMBF-Ideenwettbewerb Mensch-Technik-Interaktion in der Wissensgesellschaft:

- AUGRE - Herstellung und Wartung komplexer technischer Geräte mit Augmented-Reality Technologien und
- ASSIST - Multimodale Unterstützungssysteme für Facharbeiter der Zukunft.

AUGRE wurde im Wesentlichen vom Fraunhofer Institut für Grafische Datenverarbeitung in Darmstadt, ASSIST von der RWTH Aachen vorangetrieben.

Insbesondere die Industriepartner unterstützten die Zusammenführung der technischen Ideen dieser beiden Konsortien. So sollte sichergestellt werden, dass ausgezeichnetes Knowhow zu Basistechnologien und anerkannte Erfahrung zu Anwendungen und Nutzerorientierung als überzeugende und erfolgversprechende Projektbasis zur Verfügung stehen [Sie07b].

Als das ARVITA Projekt im Jahr 2003 zu seinem Ende kam, wurden die ersten Prototypen mobiler Augmented-Reality Systeme für den Einsatz in industriellen Anwendungen realisiert.

3.4.2 ARTESAS

Das Ziel des ARTESAS Projektes ist die Erforschung und Erprobung von Augmented-Reality Basistechnologien für den Einsatz im industriellen Service-Umfeld. Dabei setzt das Projekt auf die Ergebnisse des ARVIKA Projektes (siehe Abschnitt 3.4.1) auf. Man kann also sagen, dass es sich hierbei um die Fortsetzung des ARVIKA Projektes handelt. So sind die Projektschwerpunkte [Sie07a] ähnlich denen des ARVIKA Projektes:

- Instrumentierungsfreie Tracking-Verfahren für raue Industrieumgebungen
- Position und Orientierung des Benutzers müssen präzise erfasst werden, um die virtuelle Information lagerichtig zur realen Umgebung im Blickfeld des Nutzers einzublenden. Herausforderung ist ein universelles Framework zur anwendungsgerechten Kombination der Vorteile verschiedener Verfahren.
- Nutzergerechte AR-Geräte nach technischen und ergonomischen Gesichtspunkten

- Test und Bewertung von Geräte-Neuentwicklungen; Bereitstellung und Evaluierung einer integrierten Lösung eines AR-gerechten Head-Mounted Display mit adäquaten Wearable Komponenten.
- Umsetzung und Erprobung in industriellen Anwendungsfeldern:
    - Service für Automobile
      Informationsbereitstellung für die Automobil-Werkstatt der Zukunft - Steigerung von Effizienz und Effektivität bei Diagnose- und Instandsetzungsprozessen am Fahrzeug.
    - Service an Flugzeugen
      Vor- und Nachfluginspektion eines Hubschraubers, Reparaturszenarien an Luftfahrzeugen - Erhöhung von Produktivität und Qualität des Service. Evaluierung von AR-Technologien in engen, schwer zugänglichen Bauräumen.
    - Service in der Automatisierung
      Service-, Wartungs- und Instandhaltungsfälle in der Automatisierungstechnik - Effektivitätssteigerung durch AR-basierte Unterstützung bei unterschiedlichsten Dimensionen und Einsatzbedingungen der Anlagen (von u.a. Einzel- und Sondermaschinen bis hin zu Transferstraßen und Prozessmaschinen).

3.4.3 Liver surgery planning system

Das Liver Surgery Planing System (LSPS) wird am Institut für Computergrafik und Visualisierung an der TU Graz entwickelt. Es handelt sich dabei um ein System zur besseren Planung von Resektionen von Lebermetastasen basierend auf CT-Bilddaten. Das Hauptziel des Programmes ist es, Radiologen während der Datenaufbereitung zu unterstützen und den Chirurgen exakte Informationen zu liefern, um optimale Entscheidungen hinsichtlich der durchzuführenden Interventionen zu treffen. Dies wird durch die Kombination von medizinischer Bildanalyse und Computergrafik erreicht. Dadurch können innovative Problemlösungen gefunden werden, besonders dann, wenn eine Interaktion mit komplexen 3D-Objekten erforderlich ist [Ber07].

Mit Hilfe des LSPS ist es möglich, Resektionen zu planen und virtuelle Resektionsebenen einzuzeichnen. Außerdem können Lebergefäßbäume (siehe Abbildung 3.6) und die Volumina einzelner Lebersegmente analysiert werden.

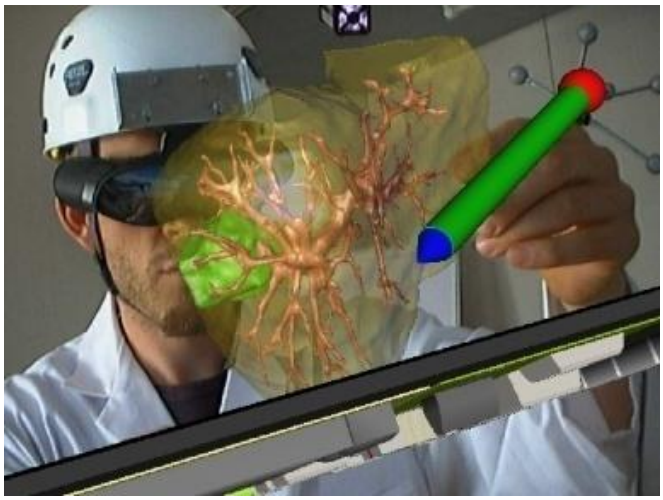

Abbildung 3.6 Augmented-Reality basiertes LSPS in Aktion [Ber07]

In Abbildung 3.7 sieht man den Aufbau des Liver Surgery Planning Systems. Es besteht aus zwei Hauptkomponenten. Zum Einen aus einem medizinischen Bildanalysesystem und zum Anderen aus dem Augmented-Reality System. Das medizinische Bildanalysesystem ist für die Segmentierung der Leber, Lebergefäßbäume und Tumore zuständig. Das Augmented-Reality System wird für die Visualisierung und die verschiedenen Arten der Benutzerinteraktion benötigt.

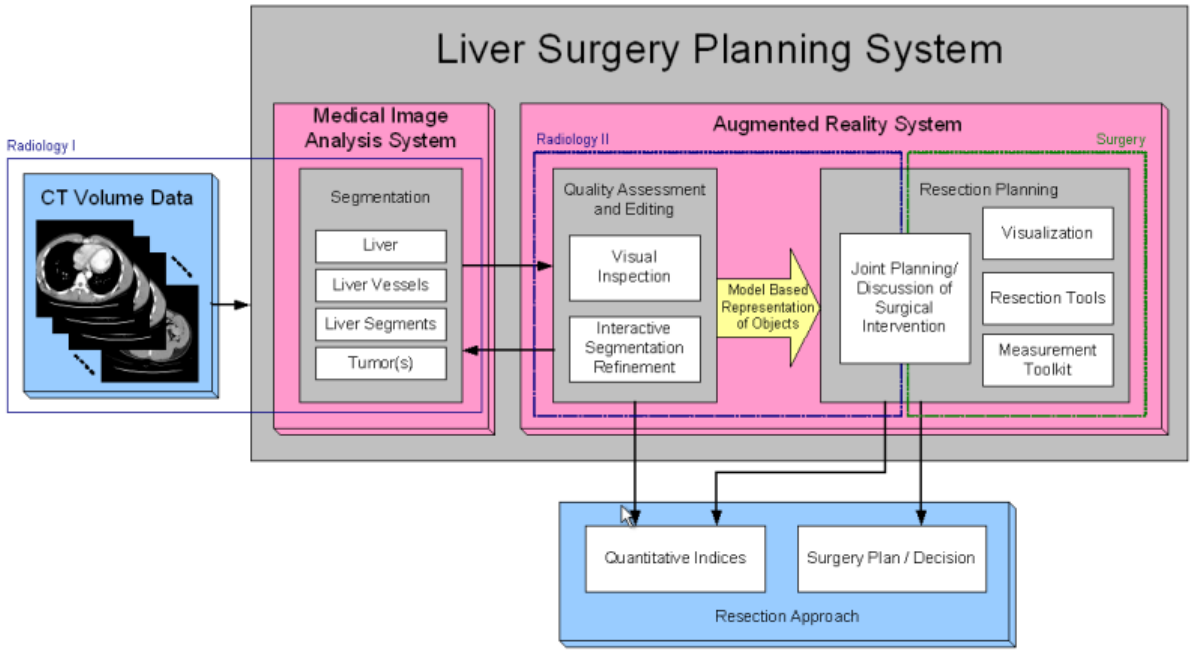

Abbildung 3.7 Überblick über die Hauptteile des Liver Surgery Planning System [Ber07]

3.4.4 MEDical Augmented-Reality for PAtients

Wie schon in Abschnitt 3.2 angedeutet, ist das Hauptproblem heutiger interventioneller Maßnahmen[4] die vielfach eingeschränkte Sicht auf den Patienten. So wird der zu behandelnde Bereich bei minimal-invasiven chirurgischen Eingriffen nur sehr begrenzt, bzw. häufig sogar gar nicht für den behandelnden Arzt sichtbar freigelegt. Somit wird der Arzt dazu gezwungen, die Operation entsprechend vorab gewonnener Daten sowie mit Hilfe seiner Erfahrungen hinsichtlich der Anatomie durchzuführen. Werden vor der Operation mehrere Modalitäten erhoben, so muss der Chirurg diese zusätzlich räumlich registrieren und in seinem Geist auf den Patienten abbilden. Ähnlich sieht es bei der Biopsie aus. Hier wird die Punktionsnadel derzeit eingeführt, ohne dabei sehen zu können, inwieweit gesunde Bereiche durch den Eingriff verletzt werden. Voraussetzung für solch einen Eingriff ist eine sorgfältige Planung sowie die Erfahrung des Arztes.

Das Ziel des Projekts MEDical Augmented-Reality for PAtients (MEDARPA) besteht darin, mit Hilfe von Augmented-Reality Technologie die Durchführung interventioneller Methoden zu unterstützen. Dabei soll mithilfe von MEDARPA die fehlende visuelle Information räumlich registriert geliefert werden, ohne dabei den Arzt durch komplizierte technische Geräte oder

[4] Diagnose- oder Therapieverfahren, die gezielte Eingriffe (Interventionen) am erkrankten Gewebe vornehmen, um den Krankheitsverlauf positiv zu beeinflussen

Aufbauten zu behindern. Somit kann der behandelnde Arzt während der Operation quasi in den Patienten hineinblicken, ohne dabei seinen Blick von der Operationsstelle abwenden zu müssen. In den meisten bisherigen Ansätzen werden Head-Mounted-Displays zur Wiedergabe genutzt. Diese sind zwar durchaus für den Laboreinsatz geeignet, sind aber aufgrund besonderer Anforderungen praktisch unbrauchbar für den realen Einsatz innerhalb eines Operationssaals. Das MEDARPA Projekt berücksichtigt diese Anforderungen und stellt die Zusatzinformation auf einem frei positionierbaren, halbtransparenten Display dar. Dieses Display kann von dem Arzt je nach Behandlungssituation optional verwendet werden. Die entwickelte Technologie lässt sich auch sehr gut auf andere Anwendungsgebiete, bei denen der Anwender auf das Tragen von zusätzlichen Geräten an seinem Körper verzichten will, übertragen (z.B. im Service und Wartungsbereich) und ist nicht nur auf den medizinischen Bereich ausgelegt [MED07].

Abbildung 3.8 Test des AR-Displays in einem Versuchsaufbau in einem Operationssaal der Uniklinik Frankfurt [MED07]

3.5 Zusammenfassung

Durch die vielen Projekte, die seit Jahren durchgeführt wurden und werden, und auch durch die Entwicklung zahlreicher Prototypen konnte nachgewiesen werden, dass Augmented-Reality eine geeignete Technologie ist, um die Bereitstellung bedarfsgerechter Informationen zu unterstützen und zu ermöglichen. Jedoch besteht auch weiterhin grundsätzlicher Forschungsbedarf, z.B. hinsichtlich der Nutzerakzeptanz und der Praxistauglichkeit mobiler Augmented-Reality Technologien. Inzwischen erkennen immer mehr Unternehmen den Wert von Augmented-Reality Technologien und treiben zusammen mit Forschungsinstituten und Universitäten die Entwicklungen auf diesem Gebiet voran. Seit der Durchführung des ARVIKA Projektes stellt Augmented-Reality einen Schwerpunkt für die Bereiche Service und Produktion dar. Es wird jedoch deutlich, dass dies stark von der Art der Realisierung der Augmented-Reality Anwendung für den jeweiligen Anwendungsfall abhängig ist und die heute verfügbaren Geräte nur zum Teil den hohen industriellen Anforderungen genügen.

Kapitel 4 Frameworks und verwendete Techniken

4.1 Bildaufnahme und Marker Erkennung

Da bei der Entwicklung von ARMediView keine neue kostenintensive Hardware angeschafft werden sollte und die Anwendung schnell und einfach auf anderen Systemen zum Einsatz kommen sollte, wurde ein optisches Tracking-System ausgewählt. Dies sollte mit Hilfe einer Videokamera oder einer Webcam und einfachen Papiermarkern erfolgen. Dadurch kann die Anwendung auf jedem PC mit einer angeschlossenen Webcam ausgeführt werden und ist nicht an spezielle Systeme und externe Hardware, wie sie z.B. bei dem Einsatz eines Systems mit mechanischer Kopplung benötigt wird, beschränkt. Es wird lediglich eine Videokamera oder Webcam sowie ein Drucker zum Erstellen der Marker benötigt.

Für das Erkennen der Marker standen mehrere Frameworks zur Auswahl. Zuerst wurde eine Sichtung vorhandener Frameworks durchgeführt. Dabei wurden sechs in die engere Wahl gezogen. Diese sollen in diesem Abschnitt näher beschrieben werden.

4.1.1 ARToolKit

Bei dem ARToolKit [Lam07] handelt es sich um eine Softwarebibliothek zur Erstellung von Augmented-Reality Anwendungen. Es berechnet die reale Kameraposition und Orientierung relativ zu den vorhandenen Markern in Echtzeit. Entwickelt wird das ARToolKit von Dr. Hirokazu Kato von der Osaka University in Japan und wird vom Human Interface Technology Laboratory (HITLab) an der University of Washington und vom HITLab NZ an der University of Canterbury in New Zealand unterstützt. Es ist für verschiedene Plattformen (Windows, Linux, Mac OS) erhältlich und komplett mit den Quellcodes kostenlos herunterlad- und einsetzbar.

Das ARToolKit bietet viele Voraussetzungen, um unter der Programmiersprache C Augmented-Reality Anwendungen zu entwickeln:

- einfaches Framework für die Erstellung Echtzeit Augmented-Reality Anwendungen
- plattformübergreifende Bibliothek (Windows, Linux, Mac OS)
- überlagert reale Marker mit virtuellen 3D Objekten (basierend auf Bildverarbeitungsalgorithmen)
- Multiplattform Bibliothek
- Unterstützung verschiedener Eingabequellen (USB, Firewire, Capture Card)
- Unterstützung verschiedener Formate (RGB/YUV420P, YUV)
- gleichzeitige Unterstützung mehrerer Kameras für das Tracking
- Interface zum Initialisieren der GUI
- schnelle und günstige 6D Marker Tracking (Echtzeit Ebenen Ermittlung)

- einfache Kalibrierung
- einfache Grafikbibliothek (basierend auf GLUT)
- schnelles Rendering basierend auf OpenGL
- VRML Unterstützung
- einfache modulare API (in C)
- Unterstützung anderer Sprachen (JAVA, Matlab)
- große Sammlung an Beispielen und Utilities
- OpenSource mit einer General Public License (GPL) für nicht-kommerzielle Nutzung

4.1.2 ARToolKitPlus

Bei ARToolKitPlus [Wag07a] handelt es sich um eine erweiterte Version des ARToolKits. Es wurde innerhalb des Handheld Augmented Reality Projekts [Wag07b] entwickelt. Im Gegensatz zum ARToolKit handelt es sich bei dem ARToolKitPlus um ein Framework, welches nicht für Augmented-Reality Anfänger geeignet ist.

Eine Auswahl an Vorteilen, die es gegenüber dem ARToolKit hat, ist:

- eine auf Klassen basierte API
- bis zu 4096 eindeutige Marker (ohne Geschwindigkeitsverlust bei einer großen Anzahl von Markern)
- neue Kameraformate (RGB565, Gray)
- variable Randbreite der Marker
- viele Geschwindigkeitsverbesserungen für einfache Endgeräte (Festkommaarithmetik, Look-Up Tabellen, etc.)
- neue Lageabschätzungsalgorithmen für stabileres Tracking (weniger Flackern)
- automatische Schwellenwertanpassung
- mehrere Tools (z.B. Marker-File nach TGA Konverter, Off-Line Kamera Kalibrierung, ID-Marker Generator)

Durch diese Änderungen eignet sich das ARToolKitPlus vor allem für den Einsatz auf mobilen Endgeräten wie PDAs oder Mobiltelefonen. Aber auch auf einem normalen Rechner machen sich die Änderungen gegenüber dem ARToolKit bemerkbar. So trägt vor allem die automatische Anpassung des Schwellwertes bei schlechten Lichtverhältnissen zu einer besseren Erkennung der Marker bei. Wie das ARToolKit wird auch das ARToolKitPlus unter der GPL veröffentlicht.

4.1.3 ARTag

Bei ARTag [ART07] handelt es sich auch um eine Software-Bibliothek für das markerbasierte Tracking. Es benutzt digitale Kodiermethoden (siehe Abbildung 4.1) anstelle der Zuordnungsmethoden wie im ARToolKit. Dies soll zu einer geringeren Fehlerrate bei der richtigen Erkennung und Zuordnung von Markern führen.

1 1 1 1 1 0
0 1 0 0 0 1
0 0 1 0 1 0
1 0 0 1 1 1
0 1 0 0 0 1
0 0 1 1 1 0

Abbildung 4.1 links: Beispiel eines Markers wie er im ARTag zum Einsatz kommt; rechts: Digitale Kodierung des Markers

Aufgrund der undurchsichtigen Lizenzbestimmungen, keinem frei zugänglichem Download, keiner Dokumentation und einer unübersichtlichen Webseite, wurde eine weitere Betrachtung von ARTag nicht durchgeführt.

4.1.4 Distributed Wearable Augmented-Reality Framework

Das Distributed Wearable Augmented-Reality Framework (DWARF) [TUM07] ist ein Framework für komponentenbasierte Peer-to-Peer Systeme. Es setzt dabei auf CORBA als Kommunikations-infrastruktur. Es basiert auf dem Konzept verteilter zusammenarbeitender Dienste. Jeder Dienst ist ein Prozess, der potentiell auf einer separaten Hardwarekomponente laufen kann und eine bestimmte Funktion anbietet oder benötigt. An jedem Netzwerkknoten eines DWARF-Systems existiert ein Service Manager. Eine zentrale Einheit existiert nicht.

4.1.5 Mixed Reality Toolkit

Bei dem Mixed Reality Toolkit [Fre07] handelt es sich um eine C++ Bibliothek, die unter der GNU Lesser General Public License veröffentlicht wurde. Das Toolkit wird von Russell Freeman am University College London entwickelt. Die aktuelle Version des Mixed Reality Toolkit ist vom 02.08.2005.

4.1.6 Das OpenIllusionist Projekt

Das OpenIllusionist Projekt [Par07] hat als Ziel, ein Framework zur Entwicklung einer Augmented-Reality Anwendung bereitzustellen. Es wird von der Media Engineering Group des Department of Electronics an der University of New York entwickelt. Es ist in C/C++ entwickelt und baut auf OpenGL auf, um alle Elemente einer OpenIllusionist-Anwendung zu rendern.

4.1.7 Zusammenfassung

In Tabelle 4.1 sind alle untersuchten Frameworks kurz zusammengefasst.

Framework	Beschreibung
ARToolKit [Lam07]	• Framework in C • Letzte Version (2.72.1) vom 02.07.2007
ARToolKitPlus [Wag07a]	• Framework in C++ • Baut auf ARToolKit auf und erweitert es um verschiedenste Funktionen
ARTag [ART07]	• Keine freie Lizenz • Aktuelle Version vom 30.11.2006
DWARF - Distributed Wearable Augmented-Reality Framework [TUM07]	• Framework für Komponentenbasierte Peer-to-Peer Systeme • baut auf CORBA als Kommunikationsplattform auf • Veröffentlicht durch die TU München
Mixed Reality Toolkit [Fre07]	• Framework in C++ • Veröffentlicht vom University College London • Benötigt DirectX 9, OpenGL, GLUT • Letzte Aktualisierung August 2005
OpenIllusionist [Par07]	• Framework in C++ • Veröffentlichet von der University of New York • Letzte Aktualisierung Juni 2006

Tabelle 4.1 Vergleich von Augmented-Reality Frameworks

Leider hat die Suche kein für eine Programmierung unter Java geeignetes Framework zutage gebracht. Daher musste ein anderer Weg gefunden werden, mit Java auf eines dieser Frameworks zurückzugreifen, da die Implementierung eines eigenen Frameworks den Rahmen dieser Arbeit sprengen würde. Zwar gibt es für das ARToolKit eine Java-Schnittstelle, um auch unter Java auf die Funktionen des ARToolKits zugreifen zu können, aber dieses ist so veraltet, dass es nicht mit der aktuellen Version des ARToolKits zusammen arbeitet. Auch die Implementation eines neuen JNI-Interfaces zur Anbindung des ARToolKitPlus, welches von den untersuchten Frameworks den größten Funktionsumfang bietet, ist aus zeitkritischen Gründen nicht möglich. Daher wurde entschieden, das jARToolKit, die Java-Schnittstelle des ARToolKits, so anzupassen und zu erweitern, dass es die aktuelle Version des ARToolKits unterstützt.

4.2 Darstellung der virtuellen Objekte

Für die Darstellung der virtuellen Objekte wurden auch wieder verschiedene Lösungen betrachtet, um so eine möglichst gute Auswahl zu tätigen. Da es sich um eine Java-Anwendung handelt, sollte für die Darstellungskomponente eine Java-Lösung gefunden werden. Daher beschränkte sich die Auswahl auch nur auf Java-APIs. Frameworks, die auf anderen Sprachen beruhen, kamen somit nicht in die engere Auswahl. So hatte auch das OpenSG-System keine Chance. Hierbei handelt es sich um ein portables Szenengraph-System zur Erstellung von Echtzeitgrafikprogrammen. Es findet sehr oft Anwendung in Virtual Reality Anwendungen.

Für Java wurden die JPCT-Engine, jMonkeyEngine und Java3D als mögliche Lösungen für das Erstellen und Rendern der virtuellen Objekte untersucht.

4.2.1 JPCT

Bei JPCT [Foe07] handelt es sich um eine 3D-Engine für Java. Sie kann sowohl in Applets als auch in Applikationen eingesetzt werden. JPCT ist ein Framework zu Programmierung von Spielen unter Java.

Mit JPCT kann die komplette virtuelle Welt und alle in ihr enthaltenen Objekte verwaltet werden. Außerdem übernimmt das Framework die komplette Darstellung dieser Welt mit allen in ihr enthaltenen Objekten.

Eigenschaften des JPCT Frameworks sind:

- objektorientiertes Design
- Kollisionserkennung zwischen den Objekten
- Unterstützung mehrerer 3D-Dateitypen (3DS, OBJ, MD2, ASC, XML) zum Laden von 3D-Objekten
- verschiedene Beleuchtungsmodi
- Wireframe (Drahtmodell) Rendering
- unterstützt sowohl Software- als auch Hardwarerendering mit Hilfe von OpenGL

4.2.2 jMonkeyEngine

Auch bei der jMonkeyEngine [Pow07] handelt es sich um ein Framework zu Erstellung von 3D-Spielen unter Java. Es ähnelt im Funktionsumfang dem von JPCT.

Eigenschaften der jMonkeyEngine:

- objektorientiertes Design
- Kollisionserkennung
- hochperformante Szenengraph basierte Grafik-API
- beliebige Rendering-Systeme lassen sich adaptieren (zur Zeit: LWJGL, in Planung JOGL)
- viele Tutorials

4.2.3 Java3D

Java3D [Col07] ist eine API zur Manipulation und Darstellung einzelner 3D-Objekte oder auch ganzer modellierter Welten innerhalb von JAVA-Applikationen und Applets. Es basiert auf DirectX™ und OpenGL und stellt somit eine Schnittstelle zu den APIs von DirectX™ und OpenGL dar. Das zentrale Konzept von Java3D ist das Szenegraph-Modell, wie es auch bei der jMonkeyEngine zum Einsatz kommt.

Bei dem Szenegraph-Modell wird jedes virtuelle Universum durch einen Szenegraphen beschrieben. Bei einem Szenegraphen handelt es sich um einen Binärbaum, bei dem die Knoten die visuellen Objekte im virtuellen Universum beschreiben (z.B. Position und Orientierung). Die Blätter stellen die Objekte selber dar.

4.2.4 Zusammenfassung

Da die zu entwickelnde Anwendung später auch als Basis für andere Anwendungen genommen werden soll, musste eine möglichst kompakte, leicht verständliche und leicht erlernbare API gewählt werden.

Unter diesen Bedingungen ist Java3D keine Lösung. Da es sich bei den anderen beiden Frameworks um sogenannte Spiele-Engines handelt, bringen diese Funktionen mit, die bei dem Einsatz von Java3D selber implementiert werden müssten. Darunter fallen z.B. die Importfunktion fertiger 3D-Modelle für verschiedene Dateiformate oder die Kollisionserkennung.

Da zwischen JPCT und der jMonkeyEngine keine gravierenden Leistungs- und Funktionsunterschiede gefunden wurden, wird die Auswahl anhand der vorhandenen Dokumentation, Tutorials, Hilfen und auch des Aufwandes zur Erstellung einer einfachen Beispiel-Applikation getroffen. Zwar ist für die jMonkeyEngine eine bessere Dokumentation vorhanden, trotzdem ließ sich mit JPCT eine Beispielanwendung einfacher und schneller realisieren. Aus diesem Grund wurde auch JPCT als Framework gewählt.

4.3 XML

Die Verwaltung der Objekte und Marker geschieht mit Hilfe der Extensible Markup Language (XML).

4.3.1 Validierung von XML-Daten

Für die Beschreibung von XML-Dateien gibt es verschiedene Möglichkeiten.

Die klassische Variante ist die Dokumenttypendefinition (engl.: Document Type Definition; kurz DTD). Sie enthält Regeln, welche Elemente in welcher Gruppierung und Häufigkeit in einem XML-Dokument verwendet werden dürfen oder müssen.

Bei XML-Schema handelt es sich um einen W3C-Standard zur Definition von XML-Dokumentgrammatiken [Spe07]. Bei XML-Schema Dokumenten handelt es sich ebenfalls um XML-Dokumente. XML-Schema ist aufgrund der größeren Möglichkeiten gegenüber DTD

wesentlich komplexer. Wie auch bei den DTDs ist das Grundprinzip der Schemadeklaration die Trennung in Elemente und Attribute. Da es mit Java einfache Möglichkeiten gibt, mit XML-Schema zu arbeiten, wurde dieses für die Definition des XML-Schematas für die XML-Dateien zur Konfiguration der 3D-Objekte und Marker verwendet.

4.3.2 XML-Parser

Um die Informationen in den XML-Dateien für die Anwendung zugänglich zu machen, gibt es verschiedene Arten von Verarbeitungsmöglichkeiten.

4.3.2.1 Document Object Model

Eine Möglichkeit bietet das *Document Object Model* (DOM). Hierbei handelt es sich um eine Entwicklung des W3C. Bei DOM wird die komplette XML-Datei eingelesen und in einer Baustruktur im Arbeitsspeicher abgelegt. Mit Hilfe von standardisierten Schnittstellen ist es nun möglich, auf die einzelnen Elemente des Dokumentenbaums wahlfrei zuzugreifen und zu bearbeiten. Dies ist auch der große Vorteil von DOM. Der Nachteil ist, dass während der gesamten Bearbeitungszeit immer das gesamte XML-Dokument im Arbeitsspeicher gehalten werden muss.

4.3.2.2 Simple API for XML Parsing

Bei *Simple API for XML Parsing* (SAX) wird das XML-Dokument nicht vollständig im Speicher abgelegt. Es basiert auf einem Ereignismodell und einem rein sequentiellen Lesen der XML-Datei. Während des Lesens der Datei werden für die einzelnen gefundenen Elemente Ereignisse ausgelöst. Dadurch ist die Verarbeitung des Dokuments sehr schnell und es wird nur ein Minimum an Arbeitsspeicher benötigt. Leider ist bei dieser Art der Verarbeitung der wahlfreie Zugriff nicht möglich.

4.3.2.3 Java Document Object Model (JDOM)

Eine weitere Möglichkeit ist das *Java Document Object Model* (JDOM). Hiermit ist es möglich, XML-Dokumente leicht und effizient mit einer schönen Java-API zu nutzen. Es basiert nicht auf DOM, auch wenn der Name dies vermuten lässt. Im Gegensatz zu SAX und DOM, die unabhängig von einer Programmiersprache sind, wurde JDOM speziell für Java entwickelt. Auch ermöglicht JDOM eine etwas bessere Performance und eine bessere Speichernutzung als beim DOM. Da mit JDOM auch eine interne Datenstruktur des XML-Dokuments erzeugt wird, kann dadurch auch jederzeit auf alle Elemente des XML-Dokuments zugegriffen werden. Da aber Java-spezifische Datenstrukturen verwendet werden, ist die Verarbeitung effizienter als bei DOM. Desweiteren ist auch eine Zusammenarbeit von JDOM und SAX möglich. So ist JDOM in der Lage, als Ausgabe SAX-Ereignisse auszulösen. So lässt sich JDOM auch sehr gut in Umgebungen einsetzen, in denen weitere Tools zur Verarbeitung von XML genutzt werden.

Auf Grund dieser vielen Vorteile wird auch für die Verarbeitung der XML-Dokumente unter ARMediView JDOM eingesetzt.

4.4 Weitere genutzte Techniken

Neben dem ARToolKit für die Bildaufnahme und Marker Erkennung, sowie dem JPCT-Framework zur Generierung und Visualisierung der virtuellen Objekte, wurden im Rahmen dieser Arbeit noch weitere Techniken verwendet. Dazu zählt die Programmiersprache Java. Für die Anbindung des ARToolKits an Java wurde JNI benutzt (siehe Abschnitt 6.4.1).

Entwickelt wurde die Anwendung mit Hilfe zweier Entwicklungsumgebungen. Auf der Java Seite kam Eclipse zum Einsatz. Da für die Anpassung des jARToolKits auch unter C einige Methoden implementiert und angepasst werden mussten, wurde dafür Microsoft Visual Studio 2005 eingesetzt. Außerdem wurde ein Subversion Repository für die Versionsverwaltung eingesetzt.

Kapitel 5 Konzept einer modularen Augmented-Reality Anwendung

Nachdem in den vorhergehenden Kapiteln ein Überblick über Augmented-Reality Systeme, Augmented-Reality Anwendungen sowie über verfügbare Hard- und Software gegeben wurde, soll in diesem Kapitel ein Konzept für eine Augmented-Reality Anwendung entworfen werden. Dabei stehen das Zusammenfügen und das Zusammenspiel der einzelnen Komponenten im Vordergrund.

Als erstes wird eine Problemanalyse durchgeführt und mögliche Lösungsansätze aufgezeichnet. Darauf aufbauend wird ein Framework für allgemeine Augmented-Reality Anwendungen entwickelt. Die Architektur dieses Frameworks wird anschließend dargestellt und die einzelnen Komponenten genauer beschrieben. Auf diesem Framework wird dann auch die Augmented-Reality Anwendung zur Darstellung medizinischer Daten aufsetzen. Aus diesem Grund wird, außer in der Analyse, im weiteren Verlauf dieses Kapitels nur noch das Framework betrachtet. Die Anwendung wird im Kapitel Kapitel 6 näher erläutert.

Im weiteren Verlauf der Arbeit, wird nicht nur von einem Marker, sondern öfters von einem Pattern gesprochen. Dabei handelt es sich um einen Teil des Markers. Ein Marker wie er in Abbildung 2.6 zu sehen ist, besteht aus einem Muster, umrahmt von einem schwarzen Rand. Bei einem Pattern handelt es sich nur um das Muster in dem Marker. Wenn im weiteren Verlauf von einem Pattern die Rede ist, so ist handelt es sich dabei meist um das Java-Objekt **`Pattern`** (siehe Abschnitt 5.3.1). Ist von einem Marker die Rede, so ist weiterhin der reale Marker gemeint.

5.1 Analyse

Im Folgenden werden die funktionalen sowie nicht funktionalen Anforderungen an das System aus der Sicht des Benutzers beschrieben und analysiert. Um diese besser zu verstehen, wird das zu lösende Problem genauer vorgestellt.

5.1.1 Problembeschreibung

Mit Hilfe der zu entwickelnden Augmented-Reality Anwendung soll ein Benutzer die Möglichkeit haben, medizinische 3D-Modelle zu betrachten und in einem gewissen Rahmen zu bearbeiten. Dabei soll der Benutzer nach Möglichkeit nicht vor einem Bildschirm sitzen, sondern so mit dem 3D-Objekt arbeiten können, als hätte er ein reales Anschauungsmodell vor sich. Ihm soll die Möglichkeit gegeben werden, das Modell zu drehen, zu skalieren und in einem gewissen Rahmen sein Aussehen zu ändern. Dabei soll die Anwendung nicht auf eine feste Basis von medizinischen Modellen beschränkt sein. Vielmehr soll diese Sammlung einfach zu erweitern und zu verwalten sein.

Die Interaktion mit dem Objekt soll dabei möglichst nicht mit Hilfe der Tastatur oder der Maus geschehen. Vielmehr soll dem Benutzer eine Eingabemöglichkeit zur Verfügung gestellt werden, die es ihm ermöglicht, das Programm zu benutzen, ohne dabei direkt vor einem Rechner sitzen zu müssen. Bei der Bedienung mit Maus und Tastatur ist der Benutzer an einen festen Arbeitsplatz gebunden, da für beide Eingabegeräte ein fester Untergrund benötigt wird. Damit ist der Bewegungsspielraum für den Benutzer stark eingeschränkt. Um diese Bindung an einen festen Arbeitsplatz zu lösen, muss eine andere Eingabemöglichkeit gefunden und entwickelt werden. Dabei soll auf eine gute Usability geachtet werden, um eine möglichst große Akzeptanz beim Anwender zu erzielen.

Zielgruppe der Anwendung sind keine Mediziner, sondern Nutzer, die sich mit medizinischen Objekten beschäftigen. Ihnen soll eine Möglichkeit gegeben werden, sich näher mit diesen Objekten befassen zu können und somit einen besseren und schnelleren Einstieg in die jeweilige Materie zu erhalten.

Neben technischen Aspekten müssen verschiedene weitere Punkte beachtet werden:

- Die Anwendung soll nicht zu komplex sein, sie muss für den Benutzer leicht verständlich und intuitiv bedienbar sein.
- Effektive Ausnutzung virtueller und realer Welt für eine Interaktion mit den virtuellen Objekten.
- Die Kosten sollen möglichst gering sein, so muss auch eine Benutzung ohne teure Head-Mounted-Displays oder ähnlicher Hardware möglich sein.
- Stabiler Arbeitsfluss auch wenn das Tracking kurzzeitig aussetzt.

5.1.2 Mögliche Lösungsansätze

Das Augenmerk bei der Entwicklung der Anwendung lag vor allem auf den letztgenannten Punkten. Die Einbeziehung des realen Raums sowie die Einfachheit der grundlegenden Bedienung standen bei der Entwicklung im Vordergrund.

Ein erster Ansatz beschränkt sich auf nur einen Marker, der entweder vor dem Benutzer auf dem Tisch liegen sollte oder vom Benutzer in die Hand genommen werden muss. Ganz so, als ob er vor einem realen Modell sitzt, welches auch in die Hand genommen werden kann. Der Nachteil bei diesem Ansatz sind die begrenzten Möglichkeiten, um das Objekt zu bearbeiten. Zwar ist es einfach, den Marker und damit auch das Objekt zu drehen. Aber spätestens beim Skalieren des Objektes stößt man an die Grenzen. Es wäre zwar möglich, den Marker näher an die Kamera, bzw. weiter weg von der Kamera zu halten, um das Objekt grösser oder kleiner zu sehen. Da es aber mit einem größer werdenden Abstand des Markers zur Kamera zu einer höheren Fehlerrate für das richtige Erkennen des Markers kommt, ist ab einer gewissen Entfernung das Erkennen des Markers nicht mehr gegeben. Bei einem Einsatz eines HMDs ist der Bewegungsspielraum des Markers auch durch die Armlänge beschränkt. Bei dieser Form der Steuerung wird das virtuelle Objekt immer an der Position des Markers angezeigt. Die beim Tracking ermittelten Positionsdaten werden unverändert auf das Objekt übertragen.

Ein weiterer Ansatz bietet in diesem Zusammenhang mehr Möglichkeiten: Bei diesem werden die Trackingdaten nicht direkt auf das virtuelle Objekt übertragen. Vielmehr soll das Objekt freischwebend im Raum dargestellt werden. Transformationen an diesem Objekt können jetzt

nicht mehr über das Bewegen des Markers durchgeführt werden. So das eine andere Möglichkeit gebraucht wird, um mit dem Objekt interagieren zu können. Daher wird ein Eingabegerät benötigt, mit dessen Hilfe das Objekt z.B. transformiert und skaliert werden kann.

Da dieser Ansatz den größten Erfolg im Bezug auf Benutzerakzeptanz verspricht, wird der erste Ansatz im Folgenden nicht weiter betrachtet.

5.1.3 Funktionale Anforderungen

Der Benutzer soll die Möglichkeit haben, mit der Anwendung Objekte zu laden, zu verschieben, zu drehen und auch wieder zu schließen. Außerdem soll es dem Benutzer möglich sein, die Auswahl an Objekten anzupassen und somit in einem gewissen Grad die Anwendung zu konfigurieren. Diese Anforderungen sind im Use Case in Abbildung 5.1 dargestellt.

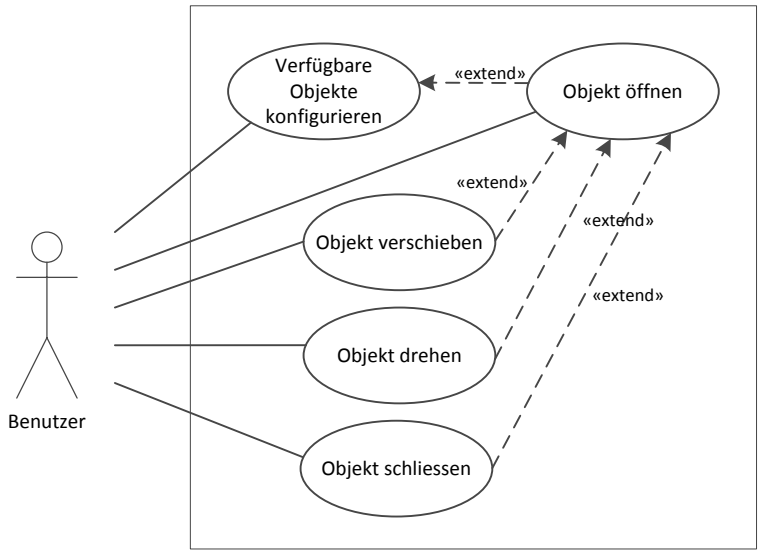

Abbildung 5.1 Use Case: Aktionsmöglichkeiten des Benutzers

5.1.4 Nicht funktionale Anforderungen

Nach [Fol94] ist die Qualität der Benutzerschnittstelle oft ausschlaggebend dafür, ob die Benutzer von einem System begeistert sind oder ob sie es ablehnen, ob sie die Entwickler des Systems verfluchen oder ob das System kommerziellen Erfolg hat. Aus diesem Grund soll auf eine Tastatur und eine Maus als Standardeingabegerät verzichtet werden, da mit diesen Geräten keine komfortable und flüssige Bedienung der Anwendung möglich ist. Aus diesem Grund muss ein Zeigegerät entwickelt werden, welches eine einfache Bedienung der Anwendung erlaubt.

Wird die Anwendung als Grundlage zur Entwicklung anderer Augmented-Reality Anwendungen benutzt, also als Framework, dann ergeben sich weitere, nicht funktionale Anforderungen. So soll dem Entwickler die Möglichkeit gegeben werden die einzelnen Komponenten möglichst einfach austauschen zu können, um somit neue Funktionen oder andere Techniken benutzen zu können. Jede Komponente für sich muss austauschbar sein, ohne dabei die anderen Komponenten ändern zu müssen.

5.2 Architektur

In diesem Abschnitt soll der Aufbau des jAR-Frameworks skizziert werden. Es werden die einzelnen Komponenten mit ihren Abhängigkeiten und Schnittstellen vorgestellt.

5.2.1 Komponenten

Das Framework besteht aus fünf Komponenten. Die Ausgangsbasis bildet die Hauptkomponente *Core*. Sie bildet die Ablaufumgebung für die folgenden Komponenten.

Die *Tracking*-Komponente kapselt die komplette Videotechnik und das Erkennen und Registrieren der Marker.

Die Komponente *Management* ist für das Laden und Verwalten der verwendeten Objekte und Marker zuständig.

Die Steuerung der Anwendung erfolgt in der Komponente *Interaction*. Sie stellt Interaktionsmöglichkeiten zur Verfügung, die z.B. durch die Marker gesteuert werden können.

Die Komponente *Display* übernimmt die Ausgabe und Darstellung der virtuellen Welt.

In Abbildung 5.2 ist der Aufbau des Frameworks grafisch dargestellt.

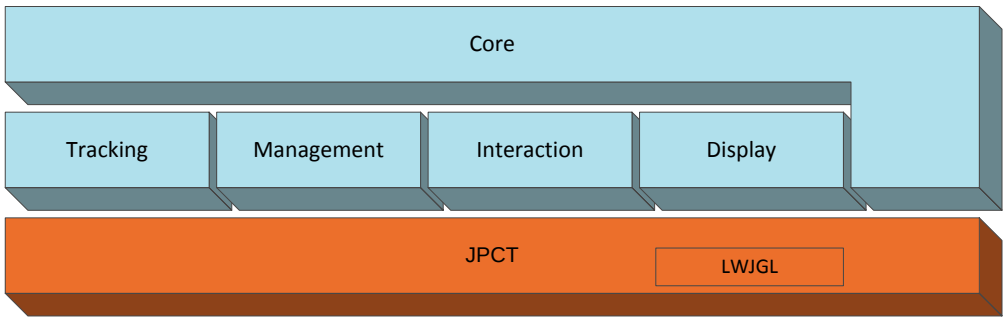

Abbildung 5.2 Komponenten des jAR-Framework

Durch die Kapselung der Implementierung der Komponenten Tracking, Management, Interaction und Display werden diese nach dem Black-Box-Prinzip aufgebaut. Durch die Schnittstellenspezifikation dieser Module werden ihre Interaktionspunkte beschrieben. Das interne Verhalten eines jeden Moduls bleibt verborgen. Damit werden nur die relevanten Informationen nach außen sichtbar, wodurch die Verständlichkeit verbessert und die Fehleranfälligkeit reduziert wird. Außerdem ist somit jederzeit der Austausch der einzelnen Komponenten möglich.

In Abbildung 5.3 sind die Ein- und Ausgaben der Komponenten vereinfacht dargestellt. Um die Aufgabe der fünf Komponenten besser verstehen zu können, wurde in Abbildung 5.4 der Datenfluss zwischen den Komponenten graphisch dargestellt.

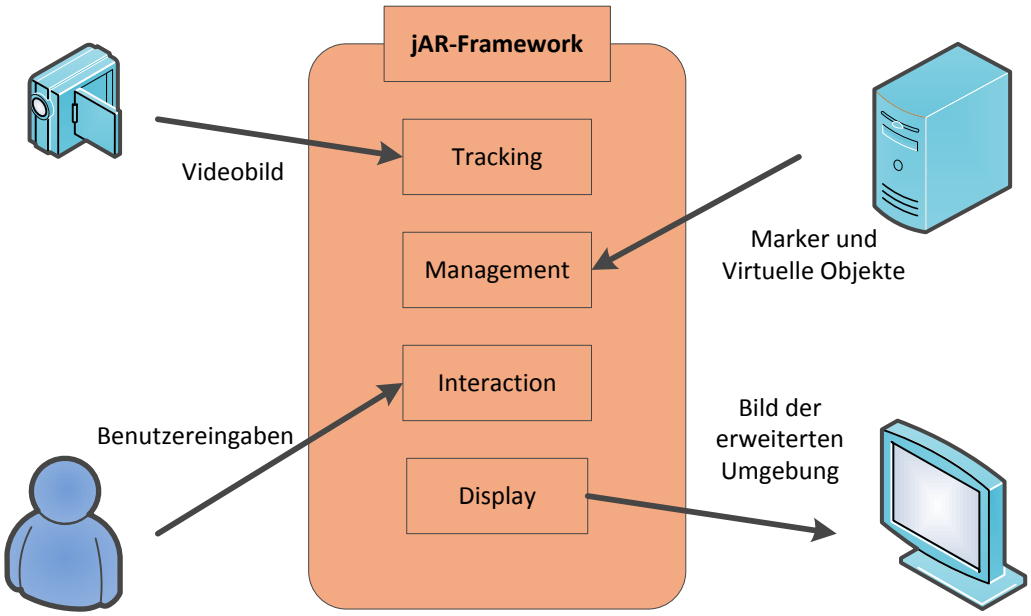

Abbildung 5.3 Ein- und Ausgabe der Komponenten

Durch die Tracking-Komponente wird ein Bild der realen Umgebung aufgenommen. In diesem werden die, von der Management-Komponente, geladenen Marker erkannt und mit Ihrer Position und Orientierung an die Management-Komponente übergeben. Die Management-Komponente liefert an die Display-Komponente die darzustellenden virtuellen Objekte. Diese rendert die komplette virtuelle Szene und überlagert damit das Bild der realen Umgebung. Durch die Interaction-Komponente kann die gesamte Anwendung beeinflusst werden. In der Abbildung ist nur die Beeinflussung von virtuellen Objekten dargestellt. Es kann aber genauso die Darstellung oder das Tracking beeinflusst werden. Der Übersichtlichkeit wegen wurde auf eine Darstellung all dieser Möglichkeiten verzichtet. Die Core-Komponente ist für den korrekten Ablauf der gesamten Anwendung zuständig. Im Konzept des jAR-Frameworks ist keine Datenverarbeitung dieser Komponente vorgesehen. Vielmehr ist die Core-Komponente für das Zusammenspiel der anderen Komponenten da. Außerdem koordiniert sie den Datenfluss zwischen den anderen Komponenten.

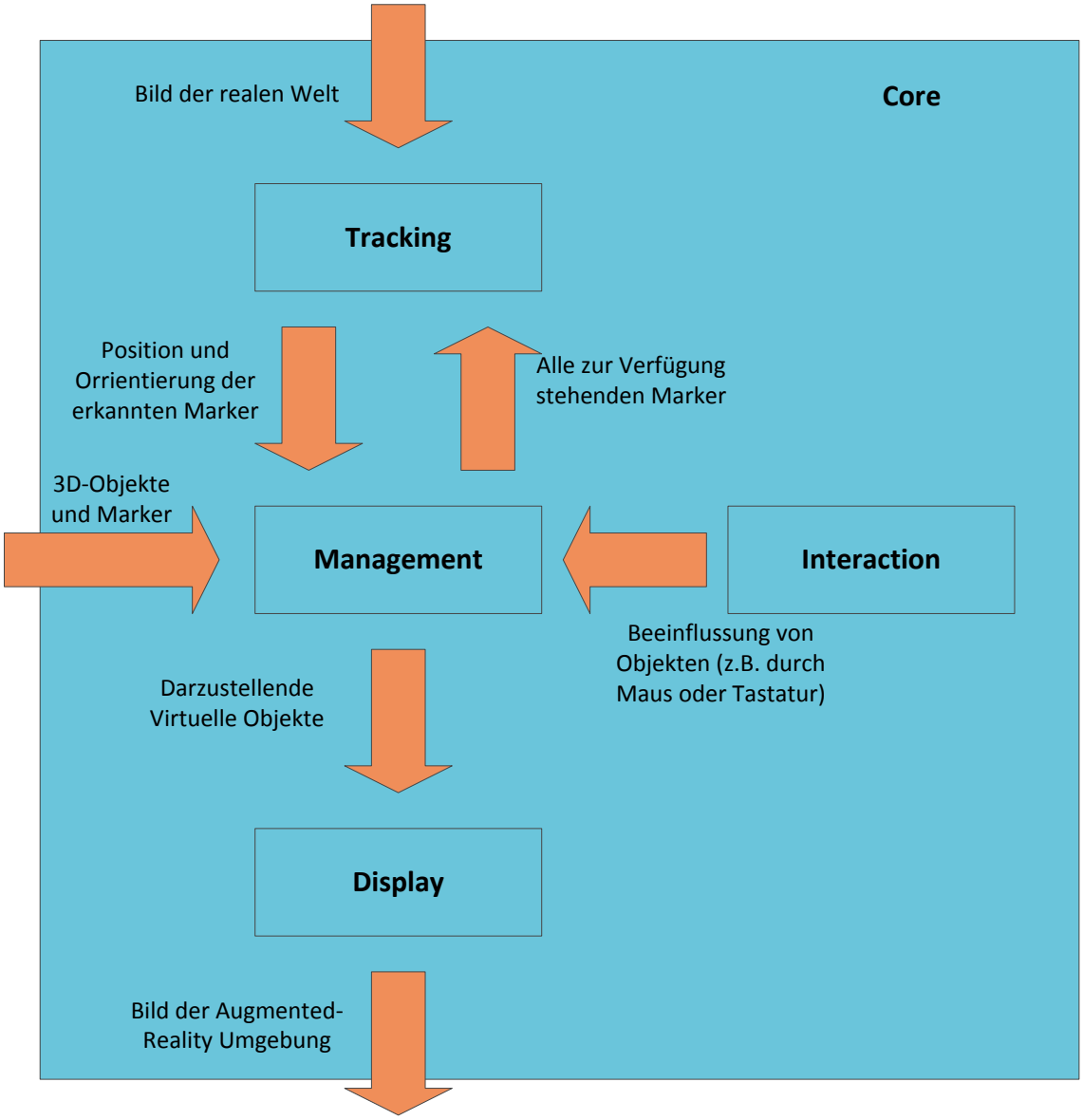

Abbildung 5.4 Datenfluss zwischen den Komponenten

5.2.2 Schichten

Die Architektur des jAR-Frameworks lässt sich in drei Schichten zerlegen:

- Präsentationsschicht
- Anwendungsschicht und
- Persitenzschicht.

In Abbildung 5.5 ist die Zugehörigkeit der einzelnen Komponenten zu den jeweiligen Schichten dargestellt.

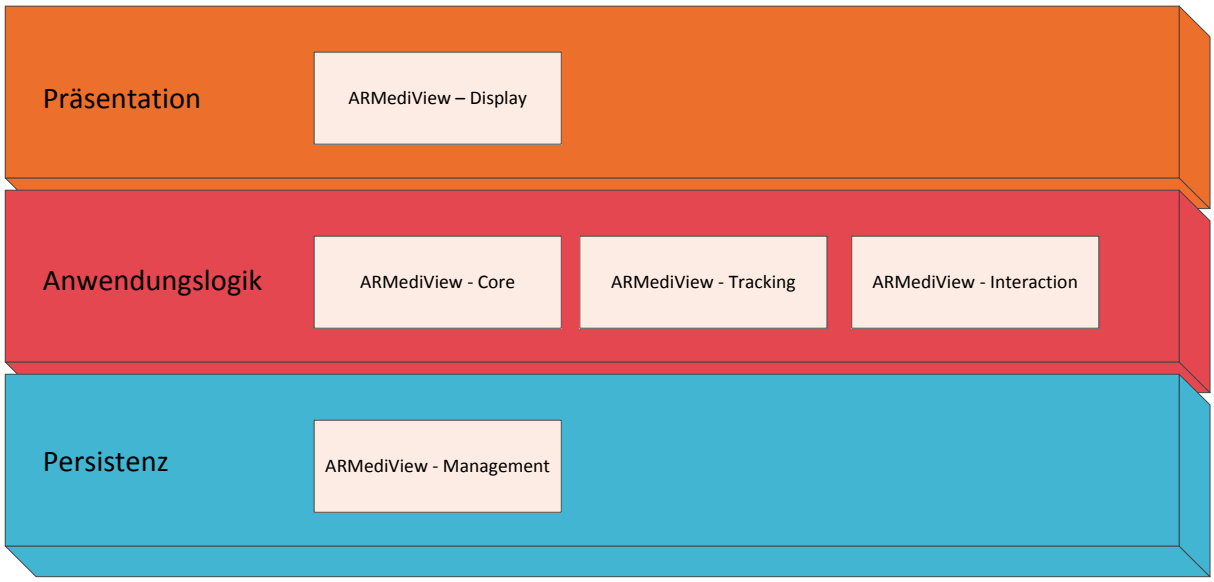

Abbildung 5.5 Schichten in der Architektur von ARMediView

5.3 Die Daten-Objekte des Frameworks

Neben den Komponenten für die einzelnen Aufgabenbereiche einer Augmented-Reality Anwendung, enthält das jAR-Framework noch zwei spezielle Objekte. Diese repräsentieren die Marker- und Virtuellen-Objekte einer Augmented-Reality Anwendung. Im Folgenden werden beide Objekte näher beschrieben. Bei beiden handelt es sich um abstrakte Objekte, da erst ihre Implementation in einer Augmented-Reality Anwendung ihren genauen Funktionsumfang definiert. Trotzdem enthalten die Objekte schon Methoden die im Folgenden beschrieben werden.

5.3.1 Das PatternObject-Objekt

Das **`PatternObject`** repräsentiert einen physikalischen Marker oder Multimarker. Das Diagramm in Abbildung 5.6 zeigt die Klasse **`PatternObject`**.

Die Klasse **`PatternObject`** kapselt alle Eigenschaften eines Markers. Diese Eigenschaften sind:

- eine eindeutige ID (wird vom Tracking-System vergeben),
- ob es sich um einen einfachen oder einen Multimarker handelt,
- die Breite des Markers,
- sein Zentrum[5],
- ob er gerade im Sichtbereich der Kamera liegt,
- sowie ob der Marker zurzeit aktiv ist.

[5] Koordinatenursprung seines Koordinatensystems von der linken unteren Ecke aus gesehen

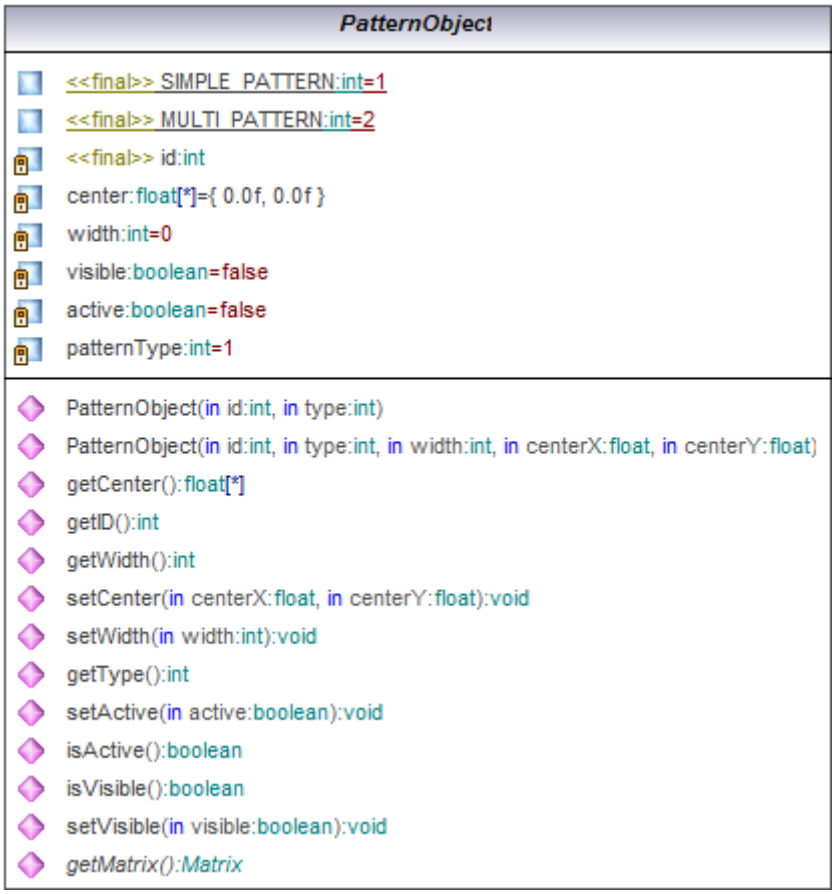

Abbildung 5.6 Die Klasse `PatternObject`

Zum Setzen und Abfragen der Eigenschaften enthält die Klasse Getter- und Setter-Methoden. Außerdem erweitert sie die Klasse **`Observable`**. Dadurch können andere Objekte auf Änderungen eines **`Pattern`**-Objektes reagieren. Um dies zu erreichen muss bei der Implementation der Klasse **`PatternObject`** mit Hilfe der Methode **`notifyObservers()`**, der Klasse **`Observable`**, die registrierten Objekte über die Änderung informiert werden. So wie die Verwendung dieser Funktionalität von der späteren Implementation der Klasse **`PatternObject`** bestimmt wird, wird auch die genaue Funktionalität der abstrakten Methode **`getMatrix()`** erst mit ihrer Implementation festgelegt.

5.3.2 Das VirtualObject-Objekt

Die Klasse **`VirtualObject`** repräsentiert ein virtuelles Objekt der Augmented-Reality Anwendung. Dazu erweitert es die Klasse **`Object3D`** aus dem JPCT-Framework. Dabei handelt es sich um eine Klasse für dreidimensionale Objekte zur Darstellung in einer virtuellen Welt. In Abbildung 5.7 ist das Diagramm der Klasse zu sehen.

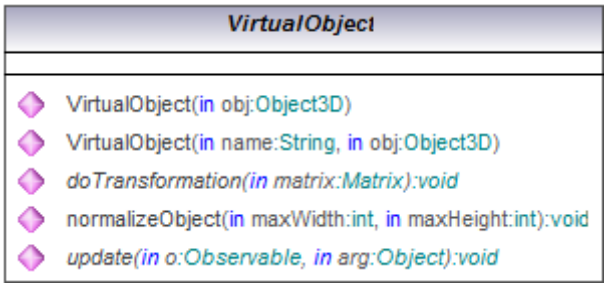

Abbildung 5.7 Die Klasse `VirtualObject`

Das **`VirtualObject`** implementiert das Interface **`Observer`**. Bei diesem handelt es sich um eine Schnittstelle, die es einem Objekt erlaubt, auf Änderungen eines Objektes vom Typ **`Observable`** zu reagieren. Dazu muss die abstrakte Methode **`update()`** implementiert werden. Somit ist es dem **`VirtualObject`** z.B. möglich auf Änderungen eines **`PatternObject`**-Objektes zu reagieren. Der abstrakten Methode **`doTransformation(…)`** wird eine Matrix übergeben. Mit Hilfe dieser kann dann z.B. das Objekt gedreht oder verschoben werden. Wie auch bei der **`update()`**-Methode bestimmt die spätere Implementation die Funktionsweise dieser Methode. Neben diesen beiden abstrakten Methoden enthält die Klasse noch die Methode **`normalizeObject(…)`**. Diese skaliert das Objekt auf eine maximale Größe. Ist das Objekt kleiner oder größer als die übergebenen Werte, wird es vergrößert oder verkleinert. Dazu werden die Abmessungen der BoundingBox[6] des Objektes und der daraus resultierende Skalierungsfaktor ermittelt und auf das Objekt angewendet.

5.4 Schnittstellen der Komponenten

Im Abschnitt 5.2 wurde das jAR-Framework als ein komponentenbasiertes Framework dargestellt. Das bedeutet, dass die einzelnen Komponenten in einer späteren Anwendung getauscht werden können. Dies ist sinnvoll wenn Teile der Anwendung angepasst werden müssen oder zum Testen anderer Technologien. Um das Einbinden anderer Komponenten zu gewährleisten, stellt das Framework in den einzelnen Komponenten keine eigene Funktionalität zur Verfügung, sondern definiert lediglich Schnittstellen. Diese stellen eine formale Deklaration der Funktionen dar, welche von der jeweiligen Komponente implementiert werden müssen. Im Folgenden werden diese Schnittstellen beschrieben. Da das Interface der Core-Komponente keine weiteren Methoden zur Verfügung stellt, wird auf dieses nicht weiter eingegangen.

5.4.1 Die Schnittstelle ITracking

Welche Trackingtechnik für die Anwendung benutzt wird, bestimmt die Implementierung des **`ITracking`**-Interfaces. Momentan gibt es eine Implementierungen, die im Abschnitt 6.4.1 näher beschrieben wird. Das Interface stellt alle Zugriffsmethoden auf das Tracking-System und die Videoquelle zur Verfügung.

---

[6] Quader, dessen Seiten parallel zu den Achsen des Objekt-lokalen Koordinatensystems stehen, und der das 3D-Objekt vollständig umschließt

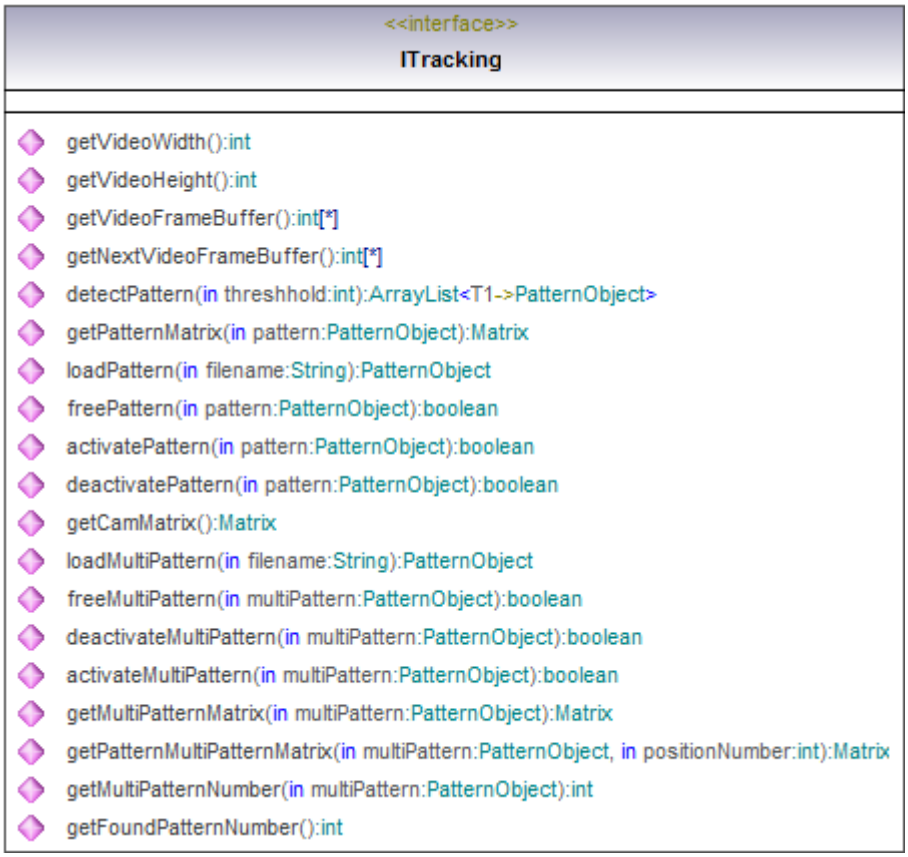

Abbildung 5.8 Diagramm der Schnittstelle `ITracking`

Abbildung 5.8 zeigt alle Methoden des Interfaces. Im Folgenden werden diese Methoden und Ihre Funktionalität dargestellt.

Da die Trackingkomponente nicht nur für das Registrieren der Marker zuständig ist, sondern auch für das Bereitstellen der Bildinformationen, können mit Hilfe der Methoden **`getVideoWidth()`** und **`getVideoHeight()`** die Größe des Videobildes ermittelt werden. Die Methode **`getNextVideoFrameBuffer()`** liefert das nächste Bild in Form eines Integer-Arrays, mit den Farbwerten von jedem Pixel des Bildes zurück. Mit **`getVideoFrameBuffer()`** ist es möglich noch einmal das aktuelle Bild zu erhalten. Mit **`getCamMatrix()`** kann die Transformationsmatrix der Kamera ermittelt werden. Um das Erkennen der Marker im aktuellen Videobild zu starten, muss der Methode **`detectPattern(...)`** ein Schwellwert für die Binärisierung des Videobildes übergeben werden. Zurückgegeben wird ein Array von **`PatternObject`**-Objekten aller erkannten Marker. Um die Marker erkennen zu können, müssen im Vorfeld mit **`loadPattern(...)`** die benutzten Marker geladen werden. Mit der Methode **`freePattern(...)`** kann ein Marker wieder entfernt werden. Um die geladenen Marker zu aktivieren, gibt es die Methode **`activatePattern(...)`**. Das Deaktivieren geschieht mit Hilfe von **`deactivatePattern(...)`**. Die Transformationsmatrix eines Markers gibt die Methode **`getPatternMatrix(...)`** zurück. Für Multimarker gibt es äquivalente Methoden. Außerdem kann die Anzahl der auf einem Multimarker enthaltenen Patterns mit der Methode **`getMultiPatternNumber(...)`** ermittelt werden. Für die einzelnen Patterns auf einem Multimarker ist es außerdem möglich, mit der Methode **`getPatternMultiPatternMatrix(...)`** die jeweilige Transformationsmatrix zu ermitteln.

5.4.2 Die Schnittstelle IManagement

Das Interface **`IManagement`** ist für die Verwaltung der Benutzten **`PatternObject`**-Objekte und **`VirtualObject`**-Objekte verantwortlich. Über verschiedene Methoden können neue Objekte hinzugefügt, vorhandene abgerufen und entfernt werden. In Abbildung 5.9 sind die von dem Interface zur Verfügung gestellten Methoden zu sehen.

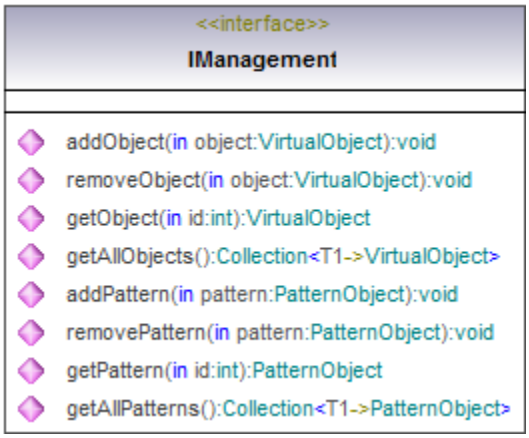

Abbildung 5.9 Diagramm der Schnittstelle `IManagment`

`addObject(...)` wird benutzt, um ein neues 3D-Objekt zur Verwaltung hinzuzufügen, um es später nutzen zu können. Mit **`removeObject(...)`** wird ein Objekt wieder aus der Verwaltung entfernt. Durch **`getObject(...)`** wird das Objekt mit der entsprechenden ID zurückgegeben. Mit Hilfe von **`getAllObjects()`** werden alle Objekte zurückgegeben. Äquivalent zu diesen Methoden für die Objektverwaltung gibt es Methoden für das Verwalten der Pattern Objekte. Mit **`addPattern(...)`** wird ein Pattern hinzugefügt. Durch **`removePattern(...)`** wird ein Pattern entfernt. **`getPattern(...)`** gibt ein Pattern, **`getAllPatterns()`** alle Patterns zurück.

5.4.3 Die Schnittstelle IInteraction

Wie in Abbildung 5.10 zu sehen, bestehet das Interface **`IInteraction`** nur aus der Methode **`poll()`**.

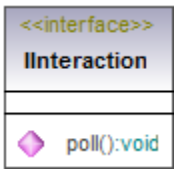

Abbildung 5.10 Diagramm der Schnittstelle `IInteraction`

Diese Methode muss aufgerufen werden um eventuelle Interaktionsevents abzuarbeiten. Sie wird nur von dem Interface zur Verfügung gestellt, um das Abfragen (engl.: poll) von Maus- und Tastaturevents anstoßen zu können. Wie dies geschieht und ob diese Methode benutzt wird oder eine andere Technik zur Abarbeitung von Maus- oder Tastaturereignissen benutzt wird, bestimmt die jeweiligen Implementation dieser Methode.

5.4.4 Die Schnittstelle IDisplay

Für die Darstellung der Augmented-Reality Umgebung ist das Interface **`IDisplay`** zuständig. In Abbildung 5.11 sind die zur Verfügung gestellten Methoden zu sehen.

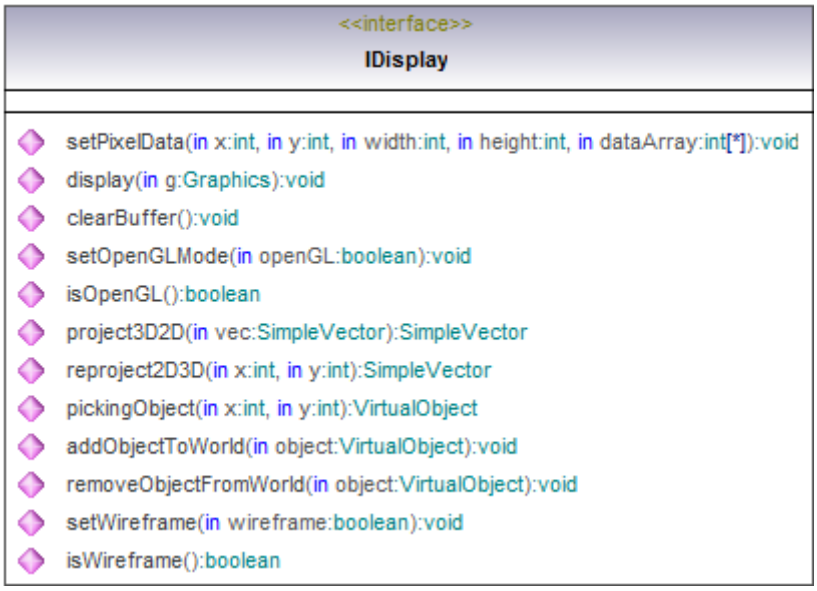

Abbildung 5.11 Diagramm der Schnittstelle `IDisplay`

Die Methode **`setPixelData(...)`** ermöglicht es, zusätzlich zu den 3D-Objekten andere Pixeldaten anzuzeigen. Dafür wird die Anfangsposition des Bildes, sowie seine Abmessungen und die Pixelinformationen übergeben. **`dispay(...)`** stellt das Ausgabebild auf dem übergebenen **`Graphics`**-Kontext dar. Mit der Methode **`clearBuffer()`** kann der FrameBuffer gelehrt werden. Die Methode **`setOpenGLMode(...)`** ermöglicht es zwischen dem OpenGL-Renderer und einem Software-Renderer umzuschalten. Der aktuelle Render-Modus kann mit Hilfe der Methode **`isOpenGL()`** ermittelt werden. Das Interface stellt außerdem zwei Methoden für die Projektion zwischen dem 3D-Kamerakoordinatensystem und 2D-Bildschirm-koordinatensystem zur Verfügung. **`project3D2D(...)`** gibt den entsprechenden Vektor im 2DKoordinatensystem zu einem übergebenen Vektor im 3D-Koordinatensystem zurück. **`reproject2D3D(...)`** liefert im Gegenzug einen Vektor des 3D-Koordinatensystems zu einer übergebenen Position im 2D-Koordinatensystem. Mit der Methode **`pickingObject(...)`** ist es möglich, zu ermitteln, welches Objekt sich an einer Position im 2D-Bildschirmkoordinaten-system befindet. Die Methode **`addObjectToWorld(...)`** fügt ein Objekt zu der virtuellen Welt hinzu. **`removeObjectFromWorld(...)`** entfernt ein Objekt aus der virtuellen Welt. Mit **`setWireframe(...)`** kann zwischen den beiden Optionen, Darstellen der kompletten virtuellen Welt als Drahtgitter (engl. Wireframe) oder mit Farbe und Texturen, umgeschaltet werden. Die Methode **`isWireframe()`** gibt den aktuellen Darstellungsmodus der Objekte zurück.

Kapitel 6 Implementierung von ARMediView

In diesem Abschnitt wird die Implementation einer Augmented-Reality Anwendung zur Darstellung von medizinischen 3D-Daten erläutert. Dabei wird das im letzten Kapitel vorgestellte Konzept und Framework als Grundlage genommen.

Da es sich bei der Anwendung um eine Augmented-Reality Anwendung zur Visualisierung medizinischer dreidimensionaler Objekte handelt, wurde die Anwendung ARMediView (Augmented-Reality Medical Data View) getauft.

6.1 Der ARMediView-Zeiger

Um eine einfache Bedienung der Anwendung zu erlaubt, wurde ein Zeigegerät entwickelt, welches in Abbildung 6.1 zu sehen ist. Dabei handelt es sich um eine Art *Zeigestock* mit dem der Benutzer die meisten Funktionen der Anwendung steuern kann. So ist es möglich, mit Hilfe des Zeigers Objekte für eine Bearbeitung zu laden, diese Objekte anschließend zu transformieren und am Ende auch wieder zu schließen.

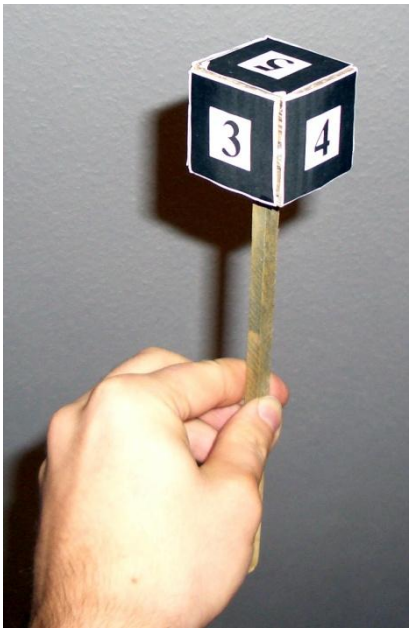

Abbildung 6.1 Beispiel eines Zeigers (ohne Tasten)

Der Zeiger besteht aus sechs einzelnen eindeutigen Markern, die zu einem Multimarker (siehe Abschnitt 2.8) zusammengefasst wurden. Dadurch ist es möglich, solange mindestens ein Marker von der Kamera erkannt wird, die genaue Lage und Orientierung des Zeigers

festzustellen. Für die Aktivierung der einzelnen Funktionen des Zeigers wurde er mit zusätzlichen Tastern ausgestattet. Die Funktionsweise des Zeigers gleicht dem einer Maus, nur dass mit Hilfe des Zeigers eine Erfassung von sechs Freiheitsgraden möglich ist, wohingegen mit einer normalen Maus[7] lediglich zwei erfasst werden können.

6.2 Die Kernkomponente von ARMediView

Für das Zusammenspiel der einzelnen Komponenten sind die Klassen im Package **`de.fhb.armediview.ARMVCore`** zuständig. Dabei handelt es sich um die Klassen **`ARMediView`**, **`AppCore`**, **`InstanceController`** und **`Utils`**. Im Folgenden soll die Funktion jeder Klasse näher beschrieben werden.

Alle Komponenten der Anwendung wurden nach dem Singleton-Pattern implementiert. Dies bedeutet, von jeder Komponente gibt es immer nur eine Instanz. Dies gilt auch für die **`AppCore`** Klasse. Mit der Klasse **`InstanceController`** wurde eine zentrale Stelle geschaffen, um Instanzen aller Komponenten bereitzustellen. Dafür gibt es für jede Komponente eine Getter-Methode, die eine Instanz der jeweiligen Komponente zurück liefert.

Die Klasse **`ARMediView`** enthält nur eine **`main`**-Methode. Durch diese wird die gesamte Anwendung gestartet. Sie holt sich vom **`InstanceController`** eine Instanz der **`AppCore`** Klasse und startet somit die Anwendung.

In der **`AppCore`** Klasse findet der eigentliche Programmablauf statt. Wird die Klasse instanziiert, so werden einige Konfigurationen der virtuellen Welt vorgenommen. Es wird z.B. die Anzahl der maximal sichtbaren Polygone festgelegt. Außerdem wird das Fenster für die Darstellung vorbereitet und die Komponente Interaction initialisiert. Sind diese Schritte abgeschlossen, so wird die Methode **`appLoop()`** aufgerufen. Diese Methode steuert den Ablauf der einzelnen Schritte einer Augmented-Reality Anwendung (siehe Abschnitt 2.6). Solange die Anwendung nicht beendet wird, wird in der **`appLoop()`**-Methode eine Endlosschleife durchlaufen. In dieser werden folgende Schritte ausgeführt:

- Aufrufen der **`poll()`** Methode der **`Interaction`** Komponente
- Speicher der Display Komponente leeren
- aktuelles Videobild der Tracking Komponente an die Display Komponente übergeben
- eine Liste aller sichtbaren Pattern ermitteln
- Sichtbarkeit aller Pattern anhand der Liste aus dem vorherigen Schritt aktualisieren
- Anzeigen des realen Bildes mit den überlagerten virtuellen Objekten durch die display Komponente.

Mehr Funktionalität wird für die Augmented-Reality Anwendung nicht benötigt. Wie, wann und wo die virtuellen Objekte angezeigt werden und wie die Anwendung auf Maus- und Tastatureingaben reagiert, wird durch die jeweiligen Komponenten gesteuert. Wie dies bei der ARMediView Anwendung funktioniert, wird im Abschnitt 6.4 erläutert.

---

[7] Es ist eine Maus ohne Scrollrad gemeint, bei einer Maus mit Scrollrad würde ein zusätzlicher Freiheitsgrad hinzukommen.

Die Klasse **`Util`** enthält mehrere Hilfsmethoden. Die Methode **`doubleArrayToMatrix(...)`** wandelt ein Array von 16 double Werten in eine Matrix um. Die Methode **`updatePatternVisibility(...)`** aktualisiert die Sichtbarkeit aller Pattern Objekte. Dazu werden alle Pattern die in der übergebenen Liste enthalten sind, als sichtbar und alle anderen als unsichtbar gekennzeichnet.

6.3 Die ARMediView-Objekte

6.3.1 Das Pattern Objekt

Die Klasse **`Pattern`** repräsentiert einen einfachen Marker. Dafür erweitert sie die abstrakte Klasse **`PatternObject`** des jAR-Frameworks. Von dieser überschreibt sie die drei Methoden **`setActive(...)`**, **`getMatrix()`** und **`setVisible(...)`**. In der Abbildung unten ist die Klasse mit ihren Methoden zu sehen.

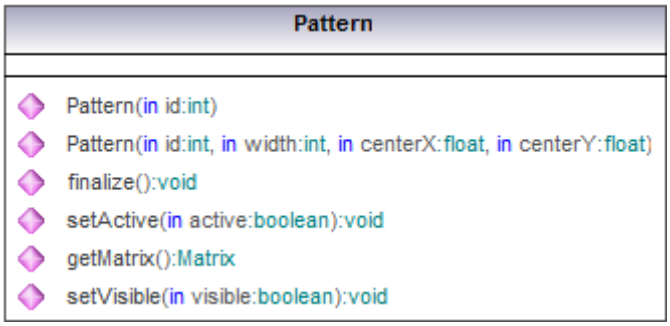

Abbildung 6.2 Die Klasse Pattern

In der **`setVisible(...)`** Methode wird mit Hilfe der Methode **`notifyObservers()`**, der Klasse **`Observable`**, die registrierten Objekte über Änderung an einem **`Pattern`**-Objekt informiert. Dies ist möglich, da die **`Pattern`**-Klasse über die **`PatternObject`**-Klasse auch die Klasse **`Observable`** erweitert. Dies geschieht immer dann, wenn der neue Zustand des Patterns sichtbar ist, egal ob er vorher auch sichtbar war oder nicht und wenn er vom Zustand sichtbar in den Zustand unsichtbar wechselt. Somit können die angemeldeten Objekte auf diese Zustandsänderung reagieren und sich z.B. auch aus- und einblenden oder an einer neuen Position angezeigt werden.

6.3.2 Das MultiPattern Objekt

Wie auch die Pattern-Klasse, repräsentiert die Klasse MultiPattern einen Marker. Nur handelt es sich hierbei nicht um einen einfachen Marker, sondern um einen Multimarker. Dafür erweitert auch sie die abstrakte Klasse **`PatternObject`** des jAR-Frameworks. Der einzige Unterschied zu der Klasse Pattern besteht in der Implementation der Methoden **`setActive(...)`** und **`getMatrix()`**. Bei der Klasse **`MultiPattern`** werden in diesen Methoden Funktionen für MultiMarker aus der Tracking-Komponente verwendet. Bei der **`Pattern`**-Klasse kommen hier Funktionen für normale Marker zum Einsatz. In der folgenden Abbildung ist die Klasse mit ihren Methoden dargestellt.

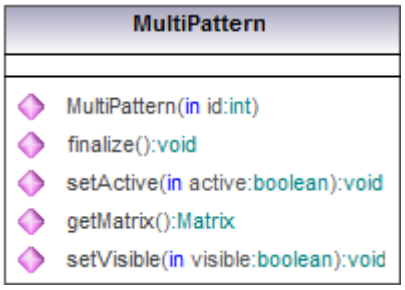

Abbildung 6.3 Die Klasse MultiPattern

6.3.3 Das ARMVObject Objekt

Die Klasse **`ARMVObject`** erweitert die abstrakte Klasse **`VirtualObject`** und repräsentiert in der Anwendung ein virtuelles Objekt. In Abbildung 6.4 ist die Klasse mit Ihren Methoden zu sehen.

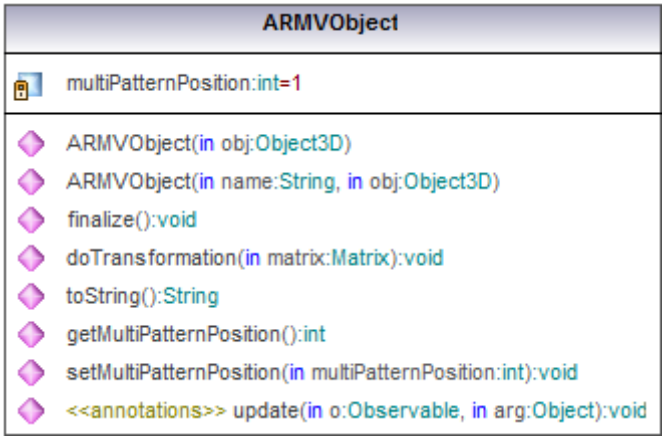

Abbildung 6.4 Die Klassen VirtualObject und PointerObject

Da die Klasse das **`Observer`**-Interface implementiert, kann sie auf Änderungen von **`Pattern`**- und **`MultiPattern`**-Objekten reagieren. Dazu werden in der **`update(…)`**-Methode Funktionen zur Anpassung der Sichtbarkeit des **`ARMVObject`**-Objekts und zur Transformation des Objektes aufgerufen. Dabei handelt es sich um die Methoden **`setVisibility(…)`** der **`Object3D`** Klasse und der **`doTransformation(…)`**-Methode. Dieser Methode wird eine Transformationsmatrix übergeben. Aus dieser wird dann eine Rotations- und eine Translationsmatrix erstellt und als aktuelle Rotations- und Translationsmatrizen des Objektes gesetzt. Das Objekt wird somit beim nächsten rendern der virtuellen Welt an der neuen Position und mit der neuen Orientierung angezeigt. Wird ein Objekt der Klasse **`ARMVObject`** im Zusammenhang mit einem Multimarker eingesetzt, so ist die Position des Objektes auf dem Multimarker durch die Eigenschaft **`multiPatternPosition`** festgelegt. Mit Hilfe der entsprechenden Getter- und Setter-Methoden kann dieser Wert geändert werden.

6.3.4 Das Pointer Objekt

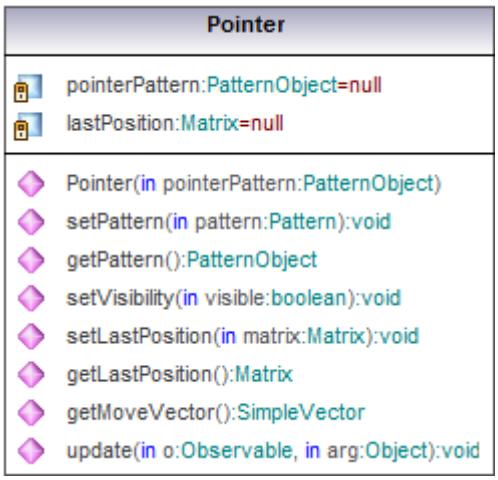

Abbildung 6.5 Die Klasse Pointer

Die Klasse **`Pointer`** repräsentiert einen Zeiger in der Augmented-Reality Umgebung. In der aktuellen Implementierung handelt es dabei um den Mauszeiger. Wie auch das **`VirtualObject`** implementiert die Klasse **`Pointer`** das Interface **`Observer`**. Dadurch kann es ebenfalls als Observer für Pattern und Multipattern Objekte dienen. Im Gegensatz zur Klasse **`VirtualObject`** existiert im **`Pointer`** ein Verweis auf das Pattern, welches zum jeweiligen Pointer gehört. Über die **`setVisibility(...)`**-Methode ist es möglich, den Mauszeiger ein und auszublenden. Dies kann notwendig sein, wenn der Marker des Pointers das Sichtfeld der Kamera verlässt. Über die update Methode ist es möglich, den Pointer auf Zustandsänderungen eines Patterns reagieren zu lassen.

6.4 Implementierung der Komponenten

In diesem Abschnitt werden die einzelnen Komponenten, die in Abschnitt 5.2.1 aufgeführt wurden, näher beschrieben.

6.4.1 ARMediView - Tracking

Mit Hilfe der Komponente *ARMediView – Tracking* kann die reale Umgebung wahrgenommen und ausgewertet werden. Sie stellt den Zugriff auf das ARToolKit zur Verfügung und ermöglicht somit die Bildaufnahme und die Marker-Erkennung durch das ARToolKit.

In Abbildung 6.6 ist der schematische Aufbau der Komponente zu sehen. Im Gegensatz zu den anderen Komponenten baut diese Komponente auf der Programmiersprache C auf. In dieser Sprache wurde das ARToolKit implementiert. Auf dieses setzt ein Wrapper auf, der alle benötigten Funktionen des ARToolKits unter Java zur Verfügung stellt. Der Zugriff auf diese Funktionen wird über eine Schnittstelle für die anderen Komponenten der Anwendung zur Verfügung gestellt.

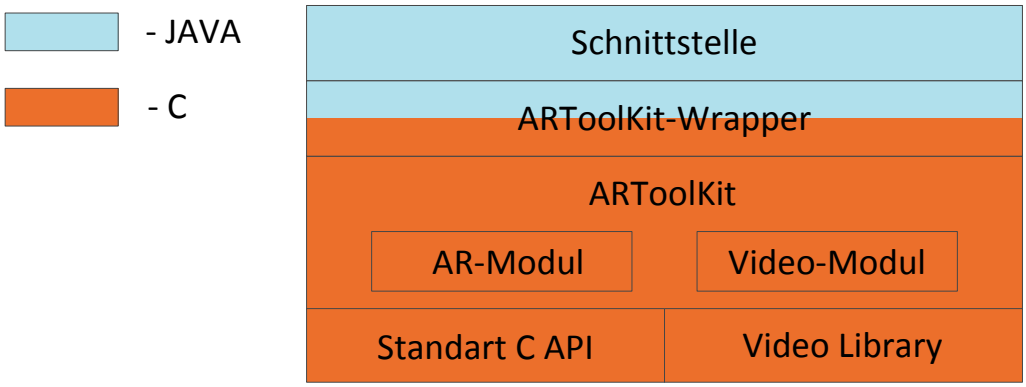

Abbildung 6.6 Aufbau der Tracking Komponente

6.4.1.1 ARToolKit

In Abschnitt 2.8 wurde bereits die Funktionsweise des ARToolKits beschrieben, aus diesem Grund soll an dieser Stelle nicht noch einmal darauf eingegangen werden. Für detailierte Informationen über die angebotenen Funktionen, sei an dieser Stelle auf die Dokumentation[8] des ARToolKits verwiesen.

6.4.1.2 jARToolKit – Die Java Schnittstelle zum ARToolKit

Um in Java auf die Funktionen des ARToolKits zugreifen zu können wird das Java Native Interface (JNI) benötigt. Bei JNI handelt es sich um eine API die es ermöglicht aus Java heraus Bibliotheken anderer Programmiersprachen und somit auch plattformspezifische Funktionen aufzurufen. Die komplette Beschreibung von JNI würde den Rahmen dieser Arbeit sprengen, daher soll hier nur ein ganz kurzer Überblick über die Funktionsweise von JNI gegeben werden. Für weitere Informationen bezüglich der Funktionsweise und der Einsatzmöglichkeiten von JNI sei hiermit auf die JNI-Seiten von Java verwiesen[9].

In Abbildung 6.7 ist der Entwicklungsprozess einer Java Anwendung, die auf C-Methoden zugreift, einfach beschrieben. In einer Java-Klasse werden die Methoden, welche die externen C-Funktionen aufrufen sollen, als **`native`** deklariert. Nach dem Kompilieren der Klasse wird mit Hilfe des Dienstprogrammes javah eine Headerdatei erzeugt. Sind alle C-Methoden implementiert und die C-Bibliothek kompiliert, wird durch den Aufruf der Java-Methode automatisch die Methode in der C-Bibliothek aufgerufen.

[8] http://artoolkit.sourceforge.net/apidoc/ [Stand: 30.09.2007]
[9] http://java.sun.com/j2se/1.4.2/docs/guide/jni/index.html [Stand: 30.09.2007]

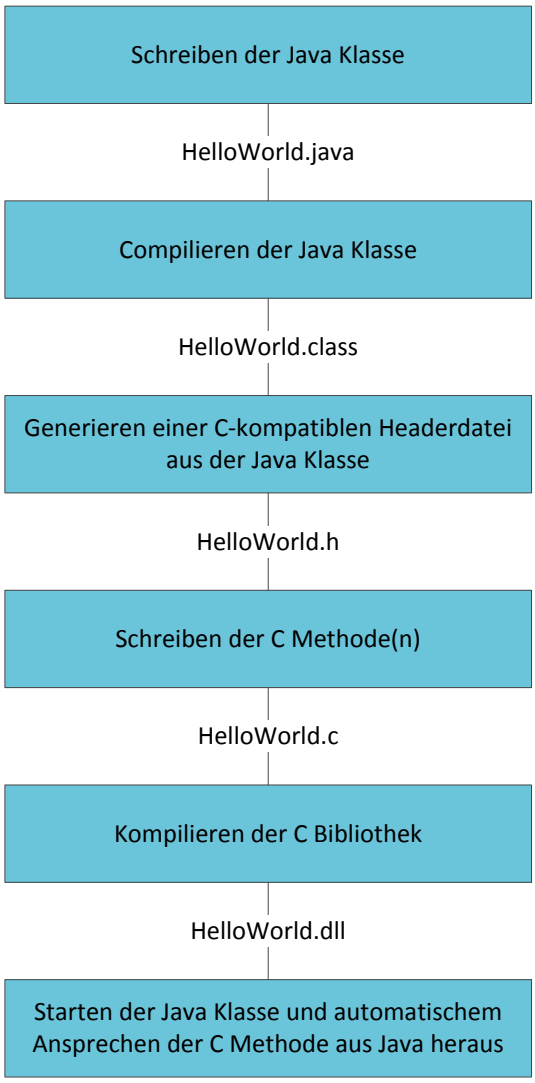

Abbildung 6.7 Schematische Darstellung eines Entwicklungsprozesses einer Java Anwendung mit Zugriff auf eine C-Bibliothek

Zwar kann mit Hilfe von JNI jede Funktion einer fremden Bibliothek genutzt werden, solange sie für andere Anwendungen nutzbar ist, aber es muss für jede Bibliothek auch eine entsprechende Methode in Java und der jeweiligen Sprache geschrieben werden. Dies ist vor allem bei großen Bibliotheken ein nicht unerheblicher Aufwand. Daher wurden auch beim jARToolKit, dem ARToolKit-Wrapper, nur die benötigten Methoden mit JNI angebunden. Es stehen somit nicht alle Funktionalitäten, die das ARToolKit bietet, zur Verfügung. Hierbei handelt es sich zum großen Teil um OpenGL- und Berechnungsfunktionen. Für diese wurden entsprechende Java-APIs benutzt oder die Funktionen wurden neu implementiert.

6.4.1.3 Die Schnittstelle zum jARToolKit

Für den Zugriff auf das jARToolKit wurde die ITracking - Schnittstelle (siehe Abschnitt 5.4.1) implementiert. Sie erlaubt einen abstrahierten Zugriff auf das jARToolKit. Da sie keine weitere Funktionalität zur Verfügung stellt, sondern lediglich den Zugriff auf die Klassen und Methoden des jARToolKits kapselt, wird die Implementierung dieser Schnittstelle hier nicht weiter beschrieben werden.

6.4.2 ARMediView – Management

Zum Laden, Initialisieren und Verwalten der Pattern- und 3D-Objekte, gibt es im Package **`de.fhb.armediview.ARMVManagement`** die Klassen **`Management`** und **`XMLLoader`**. Die Klasse **`XMLLoader`** lädt XML-Dateien und initialisiert auf deren Grundlage **`Pattern`**-, **`MultiPattern`**-, **`Pointer`** und **`VirtualObject`**-Objekte, welche die **`Management`**-Klasse für die weitere Verwendung verwaltet.

6.4.2.1 Das XML-Dokument

Die Spezifikation der Marker und Objekte erfolgt in einer XML-Datei. Je nach Anwendungsfall ist dabei pro Objekt und Marker jeweils eine XML-Datei notwendig. Mit Hilfe dieser Dateien ist es möglich, die verfügbaren Objekte und Marker in der Anwendung zu beeinflussen. Somit kann mit derselben Anwendung nur durch Austauschen der XML-Dateien auf eine völlig andere Objektauswahl zugegriffen werden. Die XML-Dateien bilden somit die Informationsbasis, die der Anwendung zur Verfügung steht.

6.4.2.2 Aufbau der XML-Datei

Im Folgenden wird das Schema (siehe Abbildung 6.9) des XML-Dokumentes beschrieben. In Abbildung 6.8 ist das Schema als Baum dargestellt.

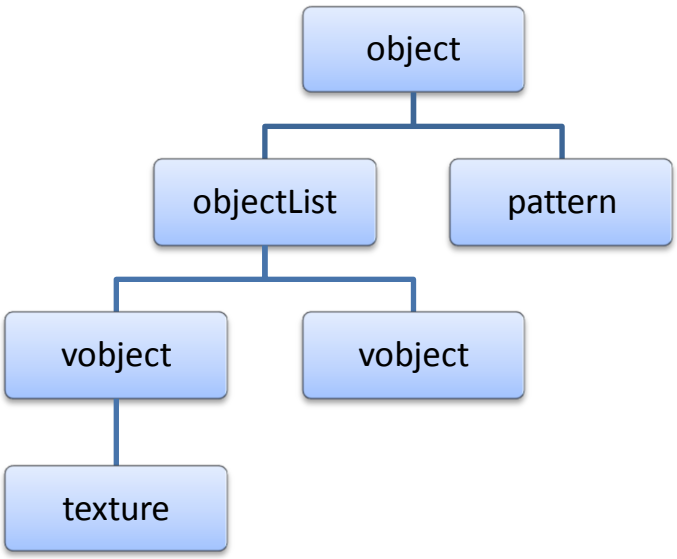

Abbildung 6.8 Struktur der XML-Datei als Baum

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

 <xsd:element name="object">
  <xsd:complexType>
   <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="objectList"/>
    <xsd:element ref="pattern"/>
   </xsd:choice>
   <xsd:attribute name="type" type="xsd:string" use="required"/>
  </xsd:complexType>
 </xsd:element>

 <xsd:element name="objectList">
  <xsd:complexType>
   <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="vobject"/>
   </xsd:choice>
  </xsd:complexType>
 </xsd:element>

 <xsd:element name="vobject">
  <xsd:complexType>
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="texture"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
   <xsd:attribute name="file" type="xsd:string" use="required"/>
   <xsd:attribute name="scale" type="xsd:double" use="required"/>
   <xsd:attribute name="rotatex" type="xsd:double" use="required"/>
   <xsd:attribute name="rotatey" type="xsd:double" use="required"/>
   <xsd:attribute name="rotatez" type="xsd:double" use="required"/>
  </xsd:complexType>
 </xsd:element>

 <xsd:element name="texture">
  <xsd:complexType>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
   <xsd:attribute name="file" type="xsd:string" use="required"/>
  </xsd:complexType>
 </xsd:element>

 <xsd:element name="pattern">
  <xsd:complexType>
   <xsd:attribute name="type" type="xsd:string" use="required"/>
   <xsd:attribute name="file" type="xsd:string" use="required"/>
   <xsd:attribute name="width" type="xsd:int"/>
   <xsd:attribute name="centerx" type="xsd:int"/>
   <xsd:attribute name="centery" type="xsd:int"/>
  </xsd:complexType>
```

Abbildung 6.9 Schema der XML-Datei

Die oberste Ebene besteht aus einem Element. Hierbei handelt es sich um das **`Object`**-Element mit dem Attribut **`type`**. Mit diesem Attribut wird die Art des Objektes festgelegt. Dabei wird unter drei verschiedenen Arten von Objekten unterschieden: **`patternObject`**, **`multiPatternObject`** und **`pointerObject`**. Die drei Arten werden im Abschnitt 6.3 näher beschrieben. Unterhalb des **`Object`**-Elements befinden sich zwei weitere Elemente **`objectList`** und **`pattern`**.

Das Element **objectList** enthält eine Menge von **vobject**-Elementen, die jeweils ein 3D-Objekt definieren. Es enthält die Attribute **name**, **file**, **scale**, **rotatex**, **rotatey** und **rotatez**. Das Attribut **name** definiert den Namen eines Objektes. Der Dateipfad wird mit dem Attribut **file** festgelegt. Es muss in einer der folgenden Formate vorliegen:

- 3DS - 3D Studio Max
- MD2 - Modellformat von Quake 2
- ASC - altes ASCII-Format von 3D-Studio
- JAW - einfaches ASCII basiertes Format der JAW-3D-Engine

Soll das Objekt eine andere Ausgangsgröße haben, so kann die gewünschte Skalierung im Attribut **scale** festgelegt werden. Mit den letzten drei Attributen **rotatex**, **rotatey** und **rotatez** kann die Ausgangsorientierung festgelegt werden. **rotatex** gibt an, um welchen Winkel das Objekt um die x-Achse gedreht werden soll. Die Attribute **rotatey** und **rotatez** werden äquivalent für die Y- und Z-Achse benutzt.

Unterhalb des **vobject**-Elements kann sich noch ein **texture**-Element befinden. Dieses legt eine Textur für das Objekt fest. Dafür enthält es zwei Attribute. Das Attribut **name**, legt den Namen der Textur fest. Über diesen kann die Textur später angesprochen werden. Das Attribut **file** gibt den Dateinamen für die Textur an.

Das **pattern**-Element definiert einen Marker. Dabei kann es sich sowohl um einen normalen Marker als auch um einen Multimarker handeln. Um was für einen Marker es sich dabei handelt, gibt das Attribut **type** an. Mögliche Werte sind **simplepattern** und **multipattern**. Das Attribut **file** gibt die Datei des Marker an. Handelt es sich um einen Multimarker, so muss hier die Konfigurationsdatei des Multimarkers angegeben werden.

bruchbedingung berechnet (hier mit einem „Vergleich“) und diese an einen „Schalter“ geführt. Ist die

In Abbildung 6.10 ist der Aufbau einer solchen XML-Datei zu erkennen. Hierbei handelt es sich um einen Multimarker mit drei dazugehörigen 3D-Objekten.

```
<?xml version="1.0" encoding="UTF-8"?>
<object type="multiPatternObject" xsi:noNamespaceSchemaLocation="armediview.xsd"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <objectList>
        <vobject name="porsche" file="porsche\porsche.3ds" scale="5"
                 rotatex="0" rotatey="0" rotatez="0" />
        <vobject name="paleta" file="paleta\PALETA.3DS" scale="5"
                 rotatex="0" rotatey="0" rotatez="0" />
        <vobject name="Bridge" file="bridge\bridge.3ds" scale="5"
                 rotatex="1.5707" rotatey="0" rotatez="0">
            <texture name="bridge" file="bridge\skin.jpg" />
        </vobject>
    </objectList>
    <pattern type="multipattern" file="multi_1\marker.dat" />
</object>
```

Abbildung 6.10 Beispiel einer XML-Datei für einen Multimarker mit drei 3D-Objekten

6.4.2.3 Zugriff auf die XML Daten

Um auf die XML-Dateien zugreifen zu können, wird der **SAXBuilder** der JDOM-API benutzt, da dieser schnell und speicherschonend arbeitet. Das XML-Dokument wird gleich beim Einlesen durch den **SAXBuilder** gegen das XML-Schema validiert. So können noch vor Beginn der Bearbeitung des XML-Dokumentes eventuell später auftretende Fehler abgefangen werden. Dies kann passieren, wenn beim Erstellen der XML-Datei nicht alle benötigten Attribute eines Objektes aufgeführt wurden, oder ein wichtiges Objekt komplett fehlt. Nach dem Validieren wird je nach Objektart (patternObject, multiPatternObject oder pointerObject) die jeweiligen **Pattern**-, **VirtualObject**- oder **Pointer**-Objekte erzeugt. Bei einem patternObject und multiPatternObject ist die Vorgehensweise gleich, nur dass bei dem patternObject ein einzelnes Objekt vom Typ **VirtualObject** erzeugt wird und bei einem multiPatternObject mehrere. Im Folgenden soll das Laden eines multiPatternObject-XML Dokument erläutert werden.

Dazu wird für jedes **vobject**-Element unterhalb des **objectList**-Elements ein neues **VirtualObject** erstellt. Dazu wird der Methode **loadObject3D()** das jeweilige **vobject**–Element übergeben. In der Methode werden die Attribute des Elementes gelesen. Durch das Laden der 3D-Datei, wird ein **VirtualObject**-Objekt erstellt. Wenn unter dem **vobject** ein **texture**-Element vorhanden ist, wird dieses Objekt noch mit einer Textur versehen. Das fertige Objekt liefert die Methode zurück. Mit dem Pattern-Element wird genauso verfahren. Es wird an die Methode **loadPattern()** übergeben. Diese Methode erzeugt dann anhand der Attribute des Elements ein **Pattern** Objekt. Sind beide Objekte erzeugt worden, so wird dem **Pattern**-Objekt das **VirtualObject**-Objekt als **Observer** hinzugefügt. Beide Objekte werden dem Manager übergeben und stehen somit der weiteren Verwendung in der Anwendung zur Verfügung.

Handelt es sich bei dem Objekt, welches das XML-Dokument beschreibt, um ein **pointerObject**, so wird nur das **Pattern**-Objekt erzeugt. Mit diesem ist es dann möglich, ein **Pointer**-Objekt zu erstellen, welches dem Pattern als **Observer** bekannt gemacht wird. Beide Objekte werden ebenfalls wie bei dem **patternObject** und dem **multiPatternObject** der Management-Komponente zur weiteren Benutzung übergeben.

6.4.3 ARMediView – Interaction

Die Komponente ARMediView – Interaction stellt Interaktionsmöglichkeiten zur Verfügung. Vom JPCT-Framework wird bereits ein KeyMapper zur Verfügung gestellt. Dieser erleichtert den Zugriff auf die Tastatur sowohl im Software- als auch im Hardwarerenderer. Für den Zugriff auf die Maus wurde ein eigener MouseMapper implementiert.

6.4.3.1 Der MouseMapper

Der MouseMapper ermöglicht es sowohl im Softwarerenderer als auch im Hardwarerender auf die Maus zuzugreifen. Um im Softwarerendermodus die Maus nutzen zu können, erweitert der MouseMapper die abstrakte Klasse MouseMotionAdapter und implementiert das MouseListener Interface. Im Hardwarerendermodus werden Funktionen der Lightweight Java Game Library (LWJGL) genutzt. An Funktionen stellt der MouseMapper neun Methoden zur Verfügung. Mit der Methode **centerMouse()** wird die Maus im Softwarerendermodus

mittig im Frame platziert. Die Methode **moveMouse(…)** bewegt die Maus an eine neue Position. Mit Hilfe von **setVisible(…)** kann der Mauszeiger ein- und ausgeblendet werden. Der aktuelle Zustand kann mit **isVisible()** abgefragt werden. Mit den Methoden **getMouseX()** und **getMouseY()** kann die aktuelle Position der Maus abgefragt werden. Um zu testen, ob eine Maustaste gedrückt wurde, kann die Methode **buttonDown(…)** benutzt werden. Übergeben werden muss der Methode die ID der jeweiligen Taste.

6.4.3.2 Die Klasse Interaction und das Abfragen des Mouse- und Keyboardmappers

Um den **Mouse**- und **KeyMapper** nutzen zu können, implementiert die Klasse Interaction das Interface IInteraction. Wie schon in Abschnitt 5.4.3 angedeutet, wird in dieser Klasse die Methode **poll()** implementiert. Sie fragt nacheinander den **KeyMapper** nach allen gedrückten Tasten und den **MouseMapper** nach evtl. gedrückten Tasten ab. Sind Tasten gedrückt, so löst die Methode entsprechende Funktionen aus. So kann in der aktuellen Implementation durch Drücken der linken Maustaste ein 3D-Objekt ausgewählt werden, auf das der Mauszeiger deutet. Dieses Objekt kann dann näher betrachtet werden und durch das Ziehen mit der Maus oder einem Pointer-Objekt verschoben und gedreht werden. Mit der rechten Maustaste wird das Objekt wieder freigegeben und ein neues Objekt kann geladen werden. Außerdem lässt sich mit verschiedenen Tasten der Tastatur das Aussehen der 3D-Objekte verändern.

6.4.4 ARMediView – Display

Die komplette Darstellung der virtuellen Objekte übernimmt die Komponente ARMediView – Display. Die Szene wird zuerst in einen **FrameBuffer** erzeugt, der sowohl einen Hardwarerenderer, in diesem Fall OpenGL, als auch einen Softwarerenderer unterstützt. Somit ist es möglich, die Ausgabe sowohl in einem Fenster zu realisieren als auch direkt in den Video-RAM der Grafikkarte zu schreiben. Dabei bleibt zu beachten das bei der aktuellen Implementierung der OpenGL-Modus nur im Zusammenhang mit einem Optical-See-Through-HMD System (siehe Abschnitt 2.3.1) zu benutzen ist, da in der aktuellen Version das Bild der realen Umgebung, im OpenGL-Modus, nicht mit gezeichnet werden kann. Wird die **display()**-Methode aufgerufen, so wird zuerst die reale Szene in den **FrameBuffer** geschrieben. Diese wird als Pixelarray von der Trackingkomponente bereitgestellt und über die **setPixelData(…)**-Methode in das **WriteableRaster** eines **BufferedImage** geschrieben, dass dann als Bild in den **FrameBuffer** gezeichnet wird. Dieses wird dann durch die gerenderten virtuellen Objekte überlagert. Das so entstandene Bild im **FrameBuffer** wird, im Softwarerendermodus, dann auf der Komponente des übergebenen Graphics-Kontextes gezeichnet. Beim Hardwarerendermodus wird der Graphics-Kontext ignoriert und das Bild in den Video-Speicher geschrieben.

Außerdem bietet die Klasse noch die Möglichkeit die BoundingBox eines Objektes auszugeben. Dadurch ist es dem Benutzer möglich, sich einen schnellen Überblick über die genauen Abmessungen des Objektes zu beschaffen.

Die Methode **pickingVirtualObject(…)** erlaubt es, Objekte zu ermitteln, die unter einem Punkt im Bildschirmkoordinatensystem liegen. Dazu wird der Vektor dieses Punktes im 3D-Koordinatensystem der virtuellen Welt ermittelt. Dieser Vektor wird als Richtungsvektor gesehen. Vom Koordinatenursprung wird dann entlang dieses Vektors das

erste Polygon, auf welches dieser Strahl trifft, ermittelt. Über die Management-Komponente kann dann das dazugehörige Objekt über die ID ermittelt werden.

Die weiteren Methoden der Klasse kapseln Funktionen, die das JPCT-Framework zur Verfügung stellt, und bieten Zugriffsfunktionen auf die Eigenschaften der Klasse.

Kapitel 7 Zusammenfassung und Ausblick

Es wurde eine Anwendung zur Darstellung von dreidimensionalen Objekten entwickelt. Das dafür erstellte Konzept ermöglicht es auf einfache Art eigene Augmented-Reality Anwendungen zu erstellen. Dafür wurde ein komponentenbasierter Ansatz ausgewählt, der es ermöglicht die einzelnen Komponenten der Anwendung auszutauschen. Dadurch ist es möglich verschiedene Tracking-, Speicher und auch Darstellungstechnologien auszuprobieren.

Im Folgenden werden Ideen für mögliche Weiterentwicklungen vorgestellt.

7.1 ARMediView als Framework

ARMediView zeigt als Framework zur Entwicklung von Augmented-Reality Anwendungen sehr viel Potential. Um es aber als eigenständiges Framework nutzen zu können, ist jedoch noch eine Weiterentwicklung notwendig. So wäre es sinnvoll nicht auf das JPCT-Framework aufzusetzen, sondern diese Funktionen zum Teil selber zu implementieren. Ein Grund dafür ist, dass es sich bei dem JPCT-Framework eigentlich um eine Spiele-Engine handelt, und sie somit doch recht überladen ist. Dies kann, vor allem bei Entwicklern, die sich in die Augmented-Reality Thematik einarbeiten wollen, zu Verwirrungen führen. Desweiteren ist die Anwendung ohne ein gutes Tracking-System nutzlos. Da bisher nur das ARToolKit mit Hilfe von JNI angebunden wurde, wäre es an dieser Stelle sinnvoll, ein eigenes Tracking-System auf JAVA Basis zu entwickeln.

Eine weitere sinnvolle Erweiterung des Frameworks wäre ein Kalibrierungstool, mit dessen Hilfe es möglich ist sowohl die Kamera, als auch die Interaktionsgeräte zu kalibrieren.

7.2 ARMediView als Anwendung

Als Anwendung zur Darstellung und Bearbeitung dreidimensionaler Objekte, ist ARMediView auch noch erweiterbar. So wäre es denkbar, neben der Betrachtung eines Objektes, auch eine erweitere Bearbeitung zu implementieren. Vorstellbar wäre das ein Objekt aufgeschnitten werden kann, oder mit anderen Objekten, zu einem großen Objekt verbunden wird. So könnte auch ein 3D-Baukasten realisiert werden, mit dem man, auf Grundlage einfacher 3D-Objekte wie z.B. einem Würfel und einer Kugel, neue komplexere Objekte zusammenbauen kann.

Literaturverzeichnis

[Fra98] Franck, Georg. *Ökonomie der Aufmerksamkeit. Ein Entwurf.* Wien : Carl Hanser, 1998.

[MC99] Milgram, Paul und Colquhoun, Herman Jr. A Taxonomy of Real and Virtual World Display Integration. [Buchverf.] Yuichi Ohta und Hideyuki Tamura. *Mixed Reality Merging Real and Virtual Worlds.* Berlin : Springer-Verlag, 1999, S. 5-30.

[Azu97] Azuma, Ronald T. *A Survey of Augmented Reality.* Malibu : Hughes Research Laboratories, 1997.

[RBG01] Rolland, Jannick P., Baillot, Yohan und Goon, Alexei A. A Survey of Tracking Technology for Virtual Environments. [Buchverf.] Woodrow Barfield und Thomas Caudell. *Fundamentals of Wearable Computers and Augmented Reality.* s.l. : Lawrence Erlbaum Associates, Inc., 2001, S. 67-112.

[Chr07] Chronos Technology Ltd. Chronos Technology. [Online] [Zitat vom: 03. 10 2007.] http://www.chronos.co.uk/pages/comp/gps-receivers.php.

[Sen07] SensAble Technologies, Inc. PHANTOM Omni - Sensable. [Online] [Zitat vom: 30. 09 2007.] http://www.sensable.com/haptic-phantom-omni.htm.

[CMT07a] CMT Consulting Measurement Technology GmbH. CMT Consulting Measurement Technology GmbH. [Online] 09. 09 2007. http://www.cmt-gmbh.de/Produkte/Inertialsysteme/AHRS500GA.html.

[CMT07b] CMT Consulting Measurement Technology GmbH. CMT Consulting Measurement Technology GmbH. [Online] [Zitat vom: 09. 30 2007.] http://www.cmt-gmbh.de/Produkte/Magnetometer/CXM539.html.

[Fia07] Fiambolis, Panos. Virtual Retinal Display (VRD) Technology. [Online] [Zitat vom: 10. 09 2007.] http://www.cs.nps.navy.mil/people/faculty/capps/4473/projects/fiambolis/vrd/vrd_full.html.

[Tho07] Pintaric, Thomas. The Invisible Train. [Online] [Zitat vom: 10. 09 2007.] http://studierstube.icg.tu-graz.ac.at/invisible_train/.

[Imm07] Immersion Corporation. Wireless Data Glove: The CyberGlove II Wireless System. [Online] [Zitat vom: 03. 10 2007.] http://www.immersion.com/3d/products/cyber_glove.php.

[Dco07] 3Dconnexion GmbH. 3Dconnexion GmbH. [Online] [Zitat vom: 3. 10 2007.] http://www.3dconnexion.de/products/3a.php.

[Ulb02] Ulbricht, Christiane. *Tangible Augmented Reality für Computerspiele.* Wien : TU Wien, 2002.

[Pae06] *Authoring von Augmented-Reality-Anwendungen.* **Paelke, Volker und Reimann, Christian.** Potsdam : KIRSCHBAUM VERLAG BONN, 2006. Kartographische Schriften. Bd. 10, S. 37-45.

[Lam07] Lamb, Philip. ARToolKit. [Online] [Zitat vom: 07. 07 2007.] http://www.hitl.washington.edu/artoolkit/.

[Sut65] *The Ultimate Display.* **Sutherland, Ivan E.** New York : s.n., 1965. Proceedings of IFIP Congress. S. 506-508.

[Sut68] *A Head-Mounted Three-Dimensional Display.* **Sutherland, Ivan.** Fall Joint Computer Conference : s.n., 1968. 1968 AFIPS Conference Proceedings. S. 757-764.

[Miz00] Mizell, David. Boeing's wire bundle assembley project. [Buchverf.] Woodrow Barfield und Thomas Caudell. *Fundamentals of Wearable Computers and Augmented Reality.* s.l. : Lawrence Erlbaum Assoc Inc., 2000, S. 447-467.

[Ruh04] Ruhe, Nikolai und Sadilek, Daniel. *MARE Augmented Reality Environment.* Berlin : s.n., 2004.

[Fuc07] Fuchs, Henry, et al. UNC Laparoscopic Visualization Research. [Online] [Zitat vom: 03. 08 2007.] http://www.cs.unc.edu/Research/us/laparo.html.

[Nat07] National University of Singapore. mixedrealitylab - Human Pacman. [Online] [Zitat vom: 15. 08 2007.] http://www.mixedreality.nus.edu.sg/index.php?option=com_content&task=view&id=42&Itemid=74.

[Pap05] Papagiannakis, George, et al. Mixing Virtual and Real scenes in the site of ancient Pompeii. *Computer Animation and Virtual Worlds.* s.l. : John Wiley and Sons Ltd., 2005, Bd. 16, S. 11-24.

[Bim02] Bimber, Oliver, et al. Merging Fossil Specimens with Computer-Generated Information. [Hrsg.] IEEE Computer Society Press. *IEEE Computer.* 09 2002, S. 45-50.

[Gra05] Grau, O. *A 3D production pipeline for special effects in TV and film.* s.l. : BBC Research & Development, 2005. White Paper WHP 108.

[Sie07b] Siemens AG. ARVIKA. [Online] 25. 07 2007. www.arvika.de.

[Sie07a] Siemens AG. ARTESAS. [Online] 01. 08 2007. www.artesas.de.

[Ber07] Bernhard, Reitinger. Virtual Liver Surgery Planning. [Online] [Zitat vom: 25. 07 2007.] http://www.icg.tu-graz.ac.at/research/interdisciplinary/liverplanner.

[MED07] MEDARPA. MEDical Augmented Reality for PAtients. [Online] 25. 07 2007. http://www.medarpa.de/.

[Wag07a] Wagner, Daniel. ARToolKitPlus. [Online] [Zitat vom: 07. 07 2007.] http://studierstube.icg.tu-graz.ac.at/handheld_ar/artoolkitplus.php.

[Wag07b] Wagner, Daniel. Handheld Augmented Reality. [Online] [Zitat vom: 07. 08 2007.] http://studierstube.icg.tu-graz.ac.at/handheld_ar/.

[ART07] ARTag. [Online] [Zitat vom: 07. 07 2007.] http://www.artag.net/.

[TUM07] TU München. DWARF - Distributed Wearable Augmented Reality Framework. [Online] [Zitat vom: 07. 07 2007.] http://campar.in.tum.de/Chair/ProjectDwarf.

[Fre07] Freeman, Russell. Mixed Reality Toolkit. [Online] [Zitat vom: 07. 07 2007.] http://www.cs.ucl.ac.uk/staff/r.freeman/index.htm.

[Par07] Parnham, Dan und Hyde, Justen. The OpenIllusionist Project. [Online] [Zitat vom: 07. 07 2007.] http://www.openillusionist.org.uk.

[Foe07] Foerster, Helge. JPCT. [Online] 24. 07 2007. http://www.jpct.net/.

[Pow07] Powell, Mark. jmonkeyengine.com. [Online] 24. 07 2007. http://www.jmonkeyengine.com/.

[Col07] CollabNet. Java3d. [Online] 24. 07 2007. https://java3d.dev.java.net/.

[Spe07] Sperberg-McQueen, C. M. und Thompson, Henry. W3C XML Schema. [Online] [Zitat vom: 18. 09 2007.] http://www.w3.org/XML/Schema.

[Fol94] Foley, James D., et al. *Grundlagen der Computergraphik.* s.l. : Addison-Wesley, 1994. 3-89319-647-1.

[Vla03] Vlahakis, V., et al. *Design and Application of an Augmented Reality System for continuous, context-sensitive guided tours of indoor and outdoor cultural sites and museums.* s.l. : VAST2003, 2003.

[Sch05a] Schmeller, Nikolaus. *Radical retropubic prostatectomy using the "Varioscope® M5".* Salzburg : University Clinic for Urology and Andrology, Paracelsus Private Medical University, 2005.

[Sch05b] Schmalstieg, Dieter. *Augmented Reality Techniques in Games.* Graz : Graz University of Technology, 2005.

[Gra04] Grau, O. *3D sequence generation from multiple cameras.* s.l. : BBC Resarch & Development, 2004. White Paper WHP 102.

[BR05] Bimber, Oliver und Raskar, Ramesh. *Spatial Augmented Reality.* Wellesley, Massachusetts : A K Peters, 2005.

[Bar06b] Barakonyi, István und Schmalstieg, Dieter. *Ubiquitous Animated Agents for Augmented Reality.* Graz : Graz University of Technology, 2006.

[Bar06a] Barakonyi, István und Schmalstieg, Dieter. *AR Puppet: Animated Agents in Augmented Reality.* Wien : Vienna University of Technology, 2006.

[Azu01] Azuma, Ronald, et al. Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications.* 2001, November / Dezember.

[ARQ07] ARQuake Project. [Online] [Zitat vom: 25. 07 2007.] http://wearables.unisa.edu.au/projects/ARQuake/www/.

Abbildungsverzeichnis

Abbildung 2.1 Einordnung von Augmented-Reality nach (2 S. 13) 4
Abbildung 2.2 Die sechs Freiheitsgrade 5
Abbildung 2.3 GPS Modul CW25-NAV der Firma Chronos Technology Ltd. (5) 6
Abbildung 2.4 PHANTOM® Omni™ Haptic Device der Firma SensAble Technologies, Inc. (6) 6
Abbildung 2.5 Inertiales Messsystem der Consulting Measurement Technology GmbH (7) 7
Abbildung 2.6 Beispiel für einen Marker wie er im ARToolKit vorkommt 8
Abbildung 2.7 Digitales High-Speed- Magnetometer CMT Consulting Measurement Technology GmbH (8) 8
Abbildung 2.8 Diagramm eines Video-See-Through-Head-Mounted-Displays nach (3) 11
Abbildung 2.9 Diagramm eines Optical-See-Through-Head-Mounted-Displays (3) 12
Abbildung 2.10 Grobes Schema eines Virtual Retinal Dispays (9) 12
Abbildung 2.11 Projektorbasierte Augmented-Reality. Links erkennt man den Versuchsaufbau, rechts die Überlagerung durch verschiedene Schattierungen und Farben 13
Abbildung 2.12 The Invisible Train (10) 14
Abbildung 2.13 CyberGlove® II Datenhandschuh der Firma Immersion Corporation (11) 15
Abbildung 2.14 SpaceNavigator der Firma 3Dconnexion GmbH (12) 15
Abbildung 2.15 Anzeige von Interaktionselementen auf einem Tablet (13) 16
Abbildung 2.16 Sicht auf die Stadt Hannover mit angereicherten Daten aus einem GIS (14) 16
Abbildung 2.17 Kamera-Koordinatensystem 17
Abbildung 2.18 Marker-Koordinatensystem 17
Abbildung 2.19 Bildschirm-Koordinatensystem 17
Abbildung 2.20 Objekt-Koordinatensystem 17
Abbildung 2.21 Beziehung zwischen Marker und Kamera Koordinatensystem 18
Abbildung 2.22 Verschiedene Koordinatensysteme und ihre Transformationen 19
Abbildung 2.23 Ablaufschema eines typischen Augmented-Reality Systems 20
Abbildung 2.24 Funktionsweise des ARToolKits 21
Abbildung 2.25 (a) das Bild der Videokamera, (b) Videobild nach Schwellwertanwendung (Schwellwert: 100) mit einem gefundenen Marker (rot markierte Fläche), (c) der gefundene Marker wird mit einem einfachen 3D-Objekt überlagert. 22
Abbildung 2.26 Multimarker bestehend aus sechs einzelnen Markern 22
Abbildung 2.27 Multimarker Konfigurationsdatei 23
Abbildung 3.1 Beispiel für die Unterstützung eines Monteurs bei der Wartung einer Waschmaschine (19 S. 24) 26
Abbildung 3.2 links: herkömmliche Operation, rechts Operation mit einem Augmented-Reality System (20). .. 27
Abbildung 3.3 Biopsie mit Hilfe von Augmented-Reality an einem Versuchsaufbau (20) 28
Abbildung 3.4 Human Pacman – Die Umgebung, durch die sich der Spieler bewegt, ist real, nur die Objekte und die überlagerten Informationen sind virtuell (21). 29
Abbildung 3.5 Anwendungsgebiete für Augmented-Reality 29
Abbildung 3.6 Augmented-Reality basiertes LSPS in Aktion (27) 31
Abbildung 3.7 Überblick über die Hauptteile des Liver Surgery Planning System (27) 32
Abbildung 3.8 Test des AR-Displays in einem Versuchsaufbau in einem Operationssaal der Uniklinik Frankfurt (28) 33
Abbildung 4.1 links: Beispiel eines Markers wie er im ARTag zum Einsatz kommt; rechts: Digitale Kodierung des Markers 37
Abbildung 5.1 Use Case: Aktionsmöglichkeiten des Benutzers 45

Abbildung 5.2 Komponenten des jAR-Framework 46
Abbildung 5.3 Ein- und Ausgabe der Komponenten 47
Abbildung 5.4 Datenfluss zwischen den Komponenten 48
Abbildung 5.5 Schichten in der Architektur von ARMediView 49
Abbildung 5.6 Die Klasse `PatternObject` 50
Abbildung 5.7 Die Klasse `VirtualObject` 51
Abbildung 5.8 Diagramm der Schnittstelle `ITracking` 52
Abbildung 5.9 Diagramm der Schnittstelle `IManagment` 53
Abbildung 5.10 Diagramm der Schnittstelle `IInteraction` 53
Abbildung 5.11 Diagramm der Schnittstelle `IDisplay` 54
Abbildung 6.1 Beispiel eines Zeigers (ohne Tasten) 55
Abbildung 6.2 Die Klasse Pattern 57
Abbildung 6.3 Die Klasse MultiPattern 58
Abbildung 6.4 Die Klassen VirtualObject und PointerObject 58
Abbildung 6.5 Die Klasse Pointer 59
Abbildung 6.6 Aufbau der Tracking Komponente 60
Abbildung 6.7 Schematische Darstellung eines Entwicklungsprozesses einer Java Anwendung mit Zugriff auf eine C-Bibliothek 61
Abbildung 6.8 Struktur der XML-Datei als Baum 62
Abbildung 6.9 Schema der XML-Datei 63
Abbildung 6.10 Beispiel einer XML-Datei für einen Multimarker mit drei 3D-Objekten 64

Tabellenverzeichnis

Tabelle 2.1 Übersicht von Trackingverfahren für Augmented-Reality 10
Tabelle 2.2 In einem AR-System vorkommende Koordinatensysteme 17
Tabelle 2.3 Zusammenhang zwischen Markergröße und Reichweite beim ARToolKit nach (15) 23
Tabelle 4.1 Vergleich von Augmented-Reality Frameworks 38

Anhang A Einstieg in die Entwicklung von Augmented-Reality Anwendungen mit Hilfe des jAR-Frameworks

Die Anleitung „Einstieg in die Entwicklung von Augmented-Reality Anwendungen mit Hilfe des jAR-Frameworks“ zur Entwicklung eigener Augmented-Reality-Anwendungen beginnt auf der nächsten Seite.

Anhang B Quelltexte und JavaDoc

Die Quelltexte des jAR-Frameworks sowie der entwickelten Anwendung ARMediView befinden sich auf der CD. Die entsprechenden JavaDoc-Dateien befindet sich ebenfalls auf der beiligenden CD.