Ruprecht-Karls-Universität Heidelberg

Hochschule Heilbronn

Diplomarbeit

März 2011

De Long Iu

Integration intuitiver Eingabegeräte ins MITK zur Optimierung von Mensch-Computer-Interaktion in der Medizin

Referent: Prof. Dr. Rolf Bendl

Koreferent: Prof. Dr. Hans-Peter Meinzer

Betreuer: Dr. Ingmar Wegner

Danksagung

Diese Arbeit wurde in der Abteilung Medizinische und Biologische Informatik (MBI) am Deutschen Krebsforschungszentrum (DKFZ) in Heidelberg durchgeführt und finanziert.

Ich danke meinem Referenten Prof. Dr. Bendl für die Betreuung dieser Arbeit. Außerdem möchte ich dem Leiter der Abteilung MBI, Prof. Dr. Hans-Peter Meinzer, für das Koreferat und die Möglichkeit die Arbeit am DKFZ anzufertigen, danken. Weiterhin gebührt meinem Betreuer Dr. Ingmar Wegner ein großer Dank für die produktiven Diskussionen und die konstruktiven Beiträge über den gesamten Verlauf der Diplomarbeit.

Ebenso vielen Dank an die gesamte Abteilung für das hervorragende Arbeitsklima und die stetige Hilfsbereitschaft. An dieser Stelle möchte ich mich vor allem bei Michi und Caspar fürs Korrekturlesen und die tatkräftige Unterstützung danken.

Der größte Dank geht an meine Familie, die mich in jeder Lebenslage unterstützt und stets an mich geglaubt hat.

Kurzfassung

Der Einsatz von virtuellen Szenen in der Medizin gewinnt zunehmend an Bedeutung, weil Navigations- und Planungshilfen für den Arzt geschaffen werden. Die Verwendung von komplexen Computersimulationen soll der Behandlungsprozess verkürzen und gleichzeitig die Behandlungsqualität durch eine gezielte Vorgehensweise verbessern. Wegen der hohen Komplexität von Bildverarbeitungsalgorithmen ist es jedoch schwierig, eine einfache Interaktion mit den medizinischen Daten zu ermöglichen. Zudem werden Benutzereingaben durch die Eingabegeräte wie Maus und Tastatur eingeschränkt, da diese in vielen Fällen keine einfache Steuerung zulassen.

Diese Diplomarbeit beschäftigt sich mit der Umsetzung intuitiver Interaktionskonzepte für den alltäglichen klinischen Gebrauch durch Verwendung von intuitiven Eingabegeräten (3D Maus, Wii Controller). Die Entwicklungen basieren auf dem Medical Imaging Interaction Toolkit (MITK) des Deutschen Krebsforschungszentrums (DKFZ). Dabei wurden von einer einfachen Kamerafahrt in einer Volumenvisualisierung über die Realisierung eines Headtracking in einer virtuellen Realität bis hin zu der Interaktion mit 3D Objekten konkrete Anwendungsbeispiele erarbeitet, analysiert und bewertet.

Inhaltsverzeichnis

1 Einleitung

Motivation

Die Verwendung von virtuellen Szenen hat sich in vielen Bereichen unseres Lebens als vorteilhaft erwiesen. Heutzutage werden Autofahrten durch Navigationssysteme unterstützt, vom digitalen Fernsehen werden zusätzliche Infos neben dem reinen Bildmaterial angeboten und in vielen technischen Bereichen lassen sich durch Computersimulationen Gefahrenpotenziale erkennen. Auch in Medizin hat diese Technologie Fuß gefasst; dort sollen Ärzte und das klinische Personal in ihrem routinierten Alltag unterstützt werden[LD05]. Aber aufgrund der hohen Komplexität der Computerverfahren und den Anforderungen an die Hardware erweist sich eine schnelle und verständliche Bedienung schwer realisierbar.

Das Hauptproblem stellt dabei die Kommunikationsschnittstelle zwischen Computer und Mensch dar. Der Einsatz von Maus und Tastatur ist durch mehrere Aspekte beschränkt:

Zunächst müssen für eine OP alle vom Arzt berührten Gegenstände einem Sterilisationsprozess unterzogen werden. Für eine chirurgische Behandlung wird also stets eine weitere Person benötigt, um die schwer sterilisierbaren Eingabegeräte (Maus und Tastatur) auf Befehl zu bedienen. In diesem Zusammenhang kommt es immer wieder zu Fehlkommunikationen, die manchmal auf die Behandlung negativen Einfluss nehmen.

Ein weiterer kritischer Aspekt ist die eigentliche Bedienung einer geeigneten Software wie das *Medical Imaging Interaction Toolkit (MITK)*. Die Entwickler achten darauf die Steuerung ihres Programms für das klinische Personal so stark wie möglich zu vereinfachen ohne dabei an Funktionalität zu verlieren. Jedoch ist diese Abstraktion nur bis zu einem gewissen Grad möglich, da jegliche Eingabe zusätzlich durch das Eingabegerät selbst eingeschränkt wird. Mit einer normalen Maus kann eine Interaktion mit einem 3D Objekt dargestellt werden, ist aber normalerweise keine leicht verständliche oder einfach ausführbare Variante. Das führt wiederum zu potentiellen Fehlbedienungen und somit zu einem erhöhten Lernaufwand durch Dokumentation und Schulungen.

Die Spieleindustrie hat dieses Problem bereits vor vielen Jahren erkannt und entwickelt benutzerfreundliche Eingabegeräte. Begonnen bei simplen Gamepads über komplexe Controller bis hin zu Steuerung durch die Gestik des Spielers selbst, haben sich auf dem Markt innovative Ideen manifestiert. Für einige Geräte wurden von Softwareingenieuren durch „Reverse Engineering“ bereits öffentlich verfügbare Treiber entwickelt. Mit deren Hilfe ist es möglich, die Ideen von Intuivität und Benutzerfreundlichkeit auf den Computer zu übertragen.

Auch im medizinischen Sektor werden erste Ansätze erprobt[FLW09]. Vor allem im Bereich der Trainingssimulation für alte sowie neue medizinische Behandlungsverfahren zu Schulungs- oder Ausbildungszwecken könnten solche Eingabegeräte signifikante Vorteile bringen.

Ziel

Das Ziel dieser Arbeit ist die Schnittstelle zwischen Computer und Mensch durch ein intuitives Eingabegerät zu ersetzen und damit in einer virtuellen Realität konkrete Aktionen auszuführen. Die dabei verwendeten Eingabegeräte sind eine 3D Maus (Space Navigator, Analog Devices) und ein Wii controller (Nintendo, Japan). Damit soll gezeigt werden, dass bereits mit einem low-cost Eingabegerät und einfachen Interaktionskonzepten die Kommunikation zur medizinischen Software MITK vereinfacht werden kann. Das reduziert wiederum den Lernaufwand und führt zu einer höheren Akzeptanz vom klinischen Personal.

Aufbau

Kapitel 2 stellt die Grundlagen für dies Arbeit vor. Dabei werden allgemein auf Interaktion, Eingabegeräte und verwendete Software eingegangen, sowie die mathematischen Grundkenntnisse für den weiteren Verlauf festgelegt.

Kapitel 3 stellt Untersuchungsergebnisse vor. Vor allem wird dort in Detail die Vorgehensweise und Umsetzung für das Integrationsdesign und die Anwendungsbeispiele hervorgehoben.

Kapitel 4 stellt eine Zusammenfassung der Ergebnisse dar. Diese werden nochmals strukturiert präsentiert und Schwerpunkte hervorgehoben.

Kapitel 5 + 6 diskutieren die Forschungsergebnisse und geben ein Ausblick auf weitere Entwicklungsmöglichkeiten.

Konventionen

Diese Arbeit ist für Leser geeignet, die Grundkenntnisse in der Informatik besitzen, da nicht jeder technischer Fachbegriff detailliert erklärt wird. Weiterhin werden englische Fachbegriffe nicht ins Deutsche übersetzt.

Quellcode wird durch die Verwendung der Schriftart Verdana vom Text abgehoben und ist entsprechend der Richtlinien von MITK in Englisch angegeben.

Neu eingeführte Begriffe werden zunächst *kursiv* geschrieben und daraufhin mit Abkürzungen oder normal weiterverwendet.

2 Material und Methoden

Dieses Kapitel stellt die Grundlage für das weitere Verständnis der Konzepte in dieser Arbeit dar. Zu Beginn wird in Kapitel 2.1 auf das allgemeine Konzept der Mensch-Computer-Interaktion eingegangen. Daraufhin werden in Kapitel 2.2 und 2.3 die technischen Details der verwendeten Eingabegeräte (Wii controller, Space Navigator) näher betrachtet. Kapitel 2.4 beschreibt die Software, auf deren Basis die Ergebnisse implementiert worden sind. Dort vorhandene Konzepte einer *virtuellen Realität (VR)*, *Augmented Reality (AR)*, *Mixed Reality (MR)* werden im folgenden Kapitel 2.5 beschrieben. Die mathematischen Grundlagen zum Thema Quaternions und Posentransformationen werden in Kapitel 2.6 und 2.7 zusammengefasst. Am Ende wird in Kapitel 2.8 ein kurzer Abriss über aktuelle Technologien gegeben.

2.1 Mensch-Computer-Interaktion

Der Begriff *Mensch-Computer-Interaktion (human-computer-interaction)* spielt eine zentrale Rolle bei der Entwicklung von Interaktionskonzepten und wird *MCI (HCI)* abgekürzt. Andreas M. Heinecke[Hei04] beschreibt dabei MCI als den wechselseitigen Prozess und die Einflussfaktoren zwischen Mensch und Computer bei einer Kommunikation. Der Benutzer (Mensch) kann eine Eingabe tätigen, auf die der Empfänger (Computer) reagiert. Daraufhin wird die Initiative umgekehrt und der Empfänger besitzt die aktive Rolle. Der Benutzer wird wiederum zur Initiative aufgefordert und kann somit Einfluss auf den weiteren Verlauf nehmen. Dieser ständige Kreislauf kann sich beliebig oft wiederholen und abhängig von der Applikation zuerst bei der aktiven Rolle des Empfängers oder Benutzers starten.

In der computerassistierten Chirurgie gibt es für diesen Bereich bislang noch wenige Untersuchungen im Bezug auf eine effiziente und korrekte Bedienung. Eine einfache und effiziente Steuerung könnte aber die Akzeptanz vom klinischen Personal erhöhen und gleichzeitig eine Reduktion der physischen und kognitiven Belastung der Benutzer bedeuten[WJ01].

Um Ergebnisse nach Messkriterien beurteilen zu können, existieren für Software Richtlinien wie die DIN EN ISO 9241. Vor allem interessant ist hier Abschnitt 11, der sich mit Gebrauchstauglichkeit beschäftigt. Dort werden drei Leitkriterien definiert:

- Effektivität zur Lösung einer Aufgabe
- Effizienz der Handhabung des Systems
- Zufriedenheit der Nutzer einer Software

Die Umsetzung solcher Eigenschaften erweist sich gerade bei computerbasierten Werkzeugen als schwierig. Traditionelle Werkzeuge wurden durch viele Entwicklungsschritte über einen langen Zeitraum an die individuellen Bedürfnisse und Präferenzen der Benutzer angepasst[Her05]. Dieser Spezialisierungsprozess befindet sich bei der Integration von Computersystemen in den klinischen Alltag erst am Anfang. Im Verlauf dieser Entwicklung könnte eine verbesserte Bedienbarkeit neue Anwendungsmöglichkeiten, wie beispielsweise realistische

Trainingssimulationen erzeugen. Der Inhalt dieser Simulationen kann eine Nachstellung einer komplexen OP sein oder auch einfache Routineverfahren zur Optimierung des Behandlungsablaufs sein. Dadurch könnte die Lernkurve für neue OP-Verfahren gesteigert werden und gleichzeitig durch beliebige Wiederholbarkeit für eine höhere Patientensicherheit gesorgt werden. Weiterhin könnte die Ausbildungsqualität gesteigert werden und im Gegenzug die Zeit für eine vollständige Schulung verringert werden[LD05].

2.2 Space Navigator

Die Firma 3DConnexion hat sich darauf spezialisiert 3D Mäuse zu entwickeln mit dem Ziel dem Benutzer eine komfortable und produktivere Eingabemöglichkeit bereitzustellen. Eine solche Maus erlaubt die Navigation in einem dreidimensionalen Raum ohne umständliche Mithilfe der Tastatur. Diese neue Sensortechnologie erlaubt laut einer Studie[Rme05] eine Reduzierung bis zu 77% des physischen Schmerzes im Gegensatz zur herkömmlichen Eingabe. Der Schmerz bei einer Mausbewegung wurde vor allem in den Bereichen: Schulter, Rücken, Nacken, Ellbogen, Handgelenk und der Hand selbst anhand des Schweregrades des subjektiven Schmerzes gemessen.

Das Model Space Navigator in der folgenden Abbildung 2.1 stellt dabei eine typische Verwendung dieser Technologie dar:

Abbildung 2.1: 3DConnexion Space Navigator[1]

[1]http://www.3dconnexion.de/products/spacenavigator.html

2.3 Wiimote

Die Spieleindustrie strebt stets nach innovativen Benutzerschnittstellen. Der Grund dafür ist ein vereinfachter Lernprozess für den Endverbraucher und die neuen Entwicklungsmöglichkeiten für Spiele. So kam es auch 2006 zur Markteinführung der Wii-Konsole von Nintendo und deren neuartigen Controller. Das Design ließ es erstmals zu über die integrierten Sensoren bis zu 5 Freiheitsgrade darzustellen. Außerdem wurde es erweitert durch eine Infrarotkamera zur Unterscheidung der einzelnen Spieler und Bestimmung von geradlinigen Bewegungen. Damit konnten komplexe Bewegungen im Raum erfasst und in die virtuelle Spielwelt abgebildet werden. Ergänzt wurde die ergonomische Form durch 12 traditionelle Gamepadtasten (A, B, 1, 2, Bewegungstasten, An/Aus Schalter, Home, +, -), einen Extension Port und eine Bluetooth-Anbindung[LJG+08].

Die folgende Abbildung 2.2 zeigt eine typische Wii-Fernbedienung.

Abbildung 2.2: Wiimote controller[2]

Der Wii-Controller wurde als *Wiimote* in der Fan-Community bekannt und erfreut sich seither großer Beliebtheit. Deswegen wurden bereits zahlreiche Modifikationen wie der Nunchuk und das Balance Board entwickelt (siehe Abbildung 2.3). Vor allem interessant ist die *Wii MotionPlus* (siehe Abbildung 2.3), die mit Hilfe der bereits vorhandenen Sensoren die Registrierung von 6 Freiheitsgraden (siehe 2.3.1) gestattet.

[2]http://www.mynintendo.de/wp-content/uploads/2006/08/wii-controller.jpg, Stand: 12-01-2011

(a) Nunchuck (b) Balance Board (c) MotionPlus

Abbildung 2.3: Extensions für die Wiimote

Das anschließende Kapitel erklärt anhand der Wiimote die 6 Freiheitsgrade. Daraufhin werden die technischen Details der normalen Wiimote und MotionPlus spezifiziert. Abschließend wird die Entwicklung von Open-Source Bibliotheken betrachtet.

2.3.1 6DoF

Für eine vollständige Abbildung eines Objektes in einem dreidimensionalen Raum sind *6 Degrees of Freedom* (6DoF) notwendig. Reza. N. Jazar[Jaz06] definiert diese Freiheitsgrade mit zwei Komponenten: Translation und Rotation.

Zunächst wird die *Translation* entlang der x-, y- und z-Achse betrachtet: Eine Translation ist eine geradlinige Bewegung in eine bestimmte Richtung. Als zweite Komponente ist die Rotation um die bereits genannten Achsen gemeint. Ein *Roll* definiert eine Rotation um die X-Achse, ein *Pitch* eine Rotation um die Y-Achse und ein *Yaw* eine Rotation um die Z-Achse. Die folgende Abbildung 2.4 zeigt diese Bewegungen im Bezug zum Wiimote Koordinatensystem.

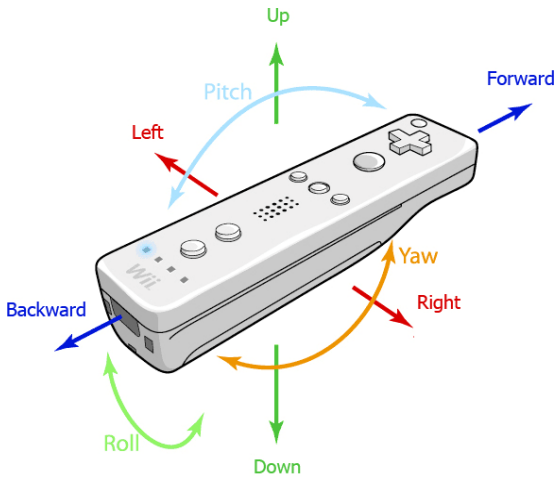


Abbildung 2.4: Koordinatenachsen in einer Wiimote[3]

[3]http://twenty7studio.ch/BA/wordpress/wp-content/uploads/2009/03/pry-wiimote.gif, Stand 16-01-2011

Durch den Treiber (siehe Kapitel 2.3.5 WiiYourself!) wird jedoch ein Roll als eine Rotation um die y-Achse und ein Pitch als eine Rotation um die x-Achse interpretiert. Der Yaw bleibt eine Rotation um die z-Achse. Die positiven oder negativen Vorzeichen bestimmen die Ausgaben der Sensoren. Eine Translation nach rechts würde also einen negativen Wert für die Beschleunigung ausgeben.

2.3.2 Infrarotkamera

In der Wiimote befindet sich eine Infrarotkamera mit einer nativen Auflösung von 128x96 Pixel. Durch ein integriertes *image processing* kann über eine Berechnung künstlich auf 1024x768 Pixel erhöht werden. Gleichzeitig können bis zur vier Infrarotquellen anhand ihrer X- und Y-Koordinaten verfolgt werden. Durch den vor der Kamera angebrachten Filter können Lichtquellen mit einer Wellenlänge von 880-940nm passieren. Abbildung 2.5 zeigt das *Field of View (FoV)*, das horizontal auf 33 Grad und vertikal auf 23 Grad beschränkt ist[4].

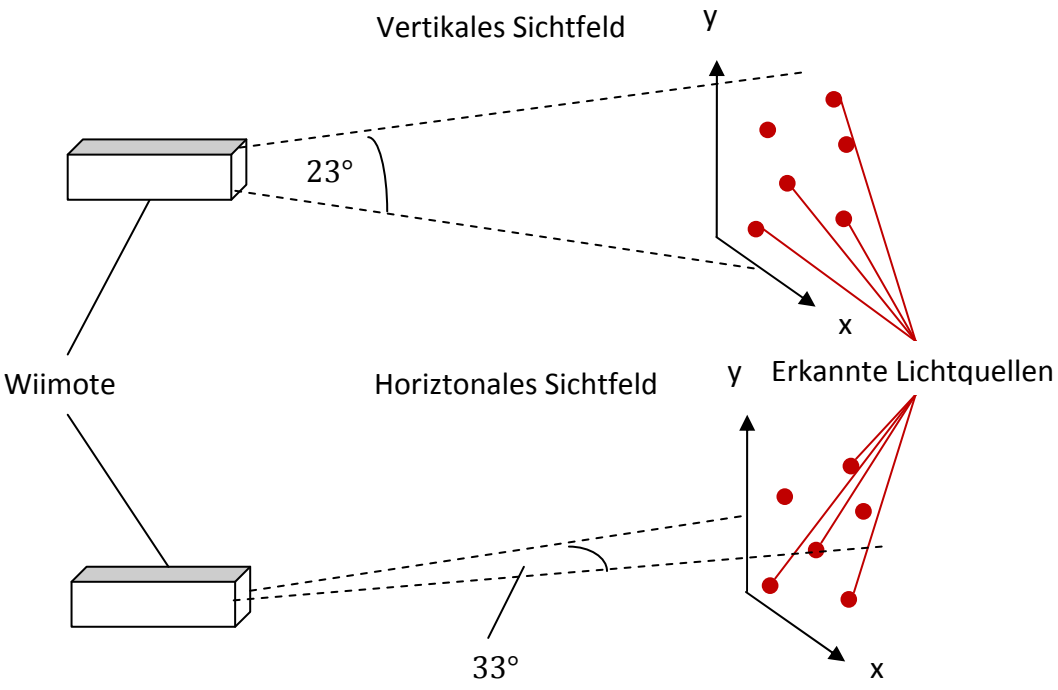


Abbildung 2.5: Sichtfeld der IR Kamera in der Wiimote

[4]http://wiibrew.org/wiki/Wiimote#IR_Camera, Stand: 12-01-2011

2.3.3 Accelerometer

Der Einsatz von *micro-electrical-mechanical systems (MEMS)* ist populär geworden und sie werden mittlerweile in Druckern, Autos und vielen anderen Bereichen verwendet. Ein spezieller Typ ist das *Accelerometer* aus dem inertialen Tracking, das über seine Sensoren lineare Beschleunigung messen kann.

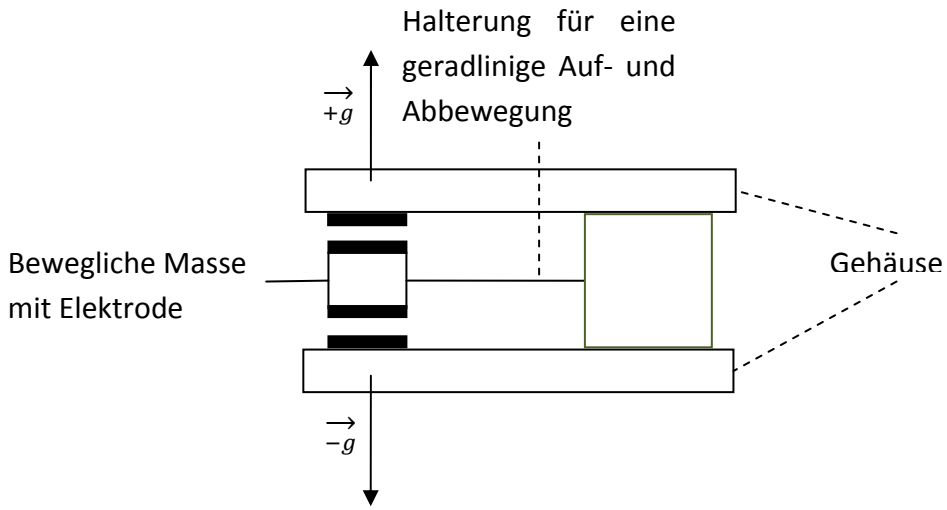


Abbildung 2.6: Skizze eines Differential-Capacitance Accelerometer[Duc05]

In der Wiimote wurde ein 3-Achsen Sensor der Kategorie *Differential-Capacitance Accelerometer* verbaut. Die Funktionsweise(siehe Abbildung 2.6) basiert auf der Registrierung der elektrischen Kapazitätsänderung aufgrund von Beschleunigungen[Duc05]. Das integrierte Model ADXL330 von Analog Devices kann dynamische Linearbeschleunigungen von -3g bis +3g und statische Beschleunigungen von -1g bis +1g erfassen[Ana07]. Die Einheit g bezieht sich auf die Wirkung der Schwerkraft auf eine Masse(siehe Formel 2.1).

$$g = 9{,}81\ \frac{m}{s^2}$$

Formel 2.1: Definition für die Einheit g

2.3.4 Gyroskop

Eine weitere Klasse der *Inertial Measurement Units (IMU)*, die oft in Verbindung mit Accelerometern genutzt wird, ist das Gyroskop[ACD+10]. Dieser Sensortyp misst die Winkelgeschwindigkeit um eine Achse durch Detektion der Corioliskraft(siehe Abbildung 2.7).

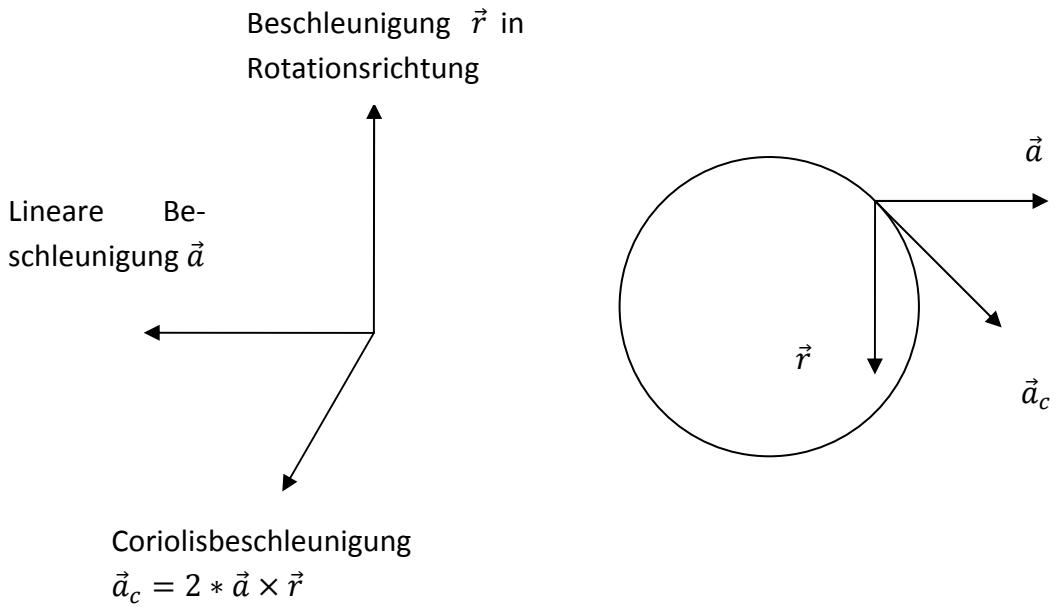


Abbildung 2.7: Wirkungsweise der Corioliskraft bei einer Rotation[TW04]

Für die Registrierung von Roll, Pitch und Yaw innerhalb der MotionPlus wurden zwei Gyroskope verbaut. Das IDG-600 von der Firma InvenSense für die X- und Y-Achse. Mit einem messbaren Intervall für die Winkelgeschwindigkeit von[Inv08]:

$$[500, 2000]\ in\ \frac{Grad}{s}$$

Formel 2.2: Intervall für Gyroskopmessbereich

Für die Z-Achse wurde ein XV-3500W von der Firma Epson Toyocom eingesetzt. Dieser Sensor ist eine Variation von dem Modell XV-3500CB von derselben Firma und wurde im Messbereich auf den IDG-600 skaliert[Eps11].

2.3.5 Bibliothek

Aufgrund der enormen Beliebtheit der Wiimote und deren einfach anzusprechende Bluetoothschnittstelle haben Entwickler durch *Reverse Engineering* eigene Treiber entwickelt. Bei diesem Prozess wurde der Controller in seine einzelnen Komponenten zerlegt und analysiert. Daraufhin wurde mit dem Gerät über die Bluetooth kommuniziert, um bestimmte Funktionalitäten anzusprechen. Dazu verwendeten die Entwickler der Open-Source Treiber USB-Geräte mit einer Bluetoothschnittstelle(Bluetoothstacks). Anhand dieser Informationen wurden eigene Treiber entwickelt mit denen Sensordaten ausgelesen, sowie Befehle zum Vibrieren der Wiimote oder Aufleuchten bestimmter LEDs umgesetzt werden konnten. Im Folgenden soll ein Überblick über die bekanntesten und funktional weit entwickelsten Bibliotheken gegeben werden [Abh09][Neß08]:

- Wiiuse/WiiuseJ
  Eine C Bibliothek, die sowohl für den Gebrauch unter Windows und Linux entworfen worden ist. Funktioniert mit den meisten Wiimotes und Bluetoothstacks. Ist lizensiert unter GNU GPLv3 und GNU LGPLv3. WiiuseJ ist eine Schnittstelle für Java zur Verwendung von Wiiuse.

- WiiRemoteJ
  Eine Java Bibliothek, die durch Java auf beliebigen Betriebssystem einsetzbar ist. Eigene Lizenz zur Weiterverwendung.

- WiiMote Lib/WiiYourself!
  WiiMote Lib ist ein C# und VB.NET Bibliothek, die nur unter Windows funktioniert. Sie bietet Grundfunktionalitäten für die Nutzung der Wiimote. Lizensiert unter der Microsoft Public License.
  WiiYourself! basiert auf der oben genannten Bibliothek, wurde jedoch um Funktionalität erweitert und in C++ programmiert. Volle Unterstützung aller Wiimotes, Bluetoothstacks und Zugriffsmöglichkeiten auf Hardware. Eigene Lizenz zur Weiterverwendung.

2.4 Softwarekomponenten

Um mit medizinschen Daten interagieren zu können, benötigt der Benutzer eine passende Software. Die Softwarebibliothek MITK zeichnet sich dabei durch seine Vielfältigkeit und Interaktionskonzepte aus. Im Folgenden werden die dort verwendeten Toolkits sowie Konzepte erläutert, die von Bedeutung für diese Arbeit sind.

2.4.1 MITK

Das Medical Imaging Interaction Toolkit (MITK) ist eine plattformunabhängige C++ Software-Bibliothek (siehe Abbildung 2.8), die seit 2002 vom Deutschen Krebsforschungszentrum (DKFZ) entwickelt wird[MMS+10].

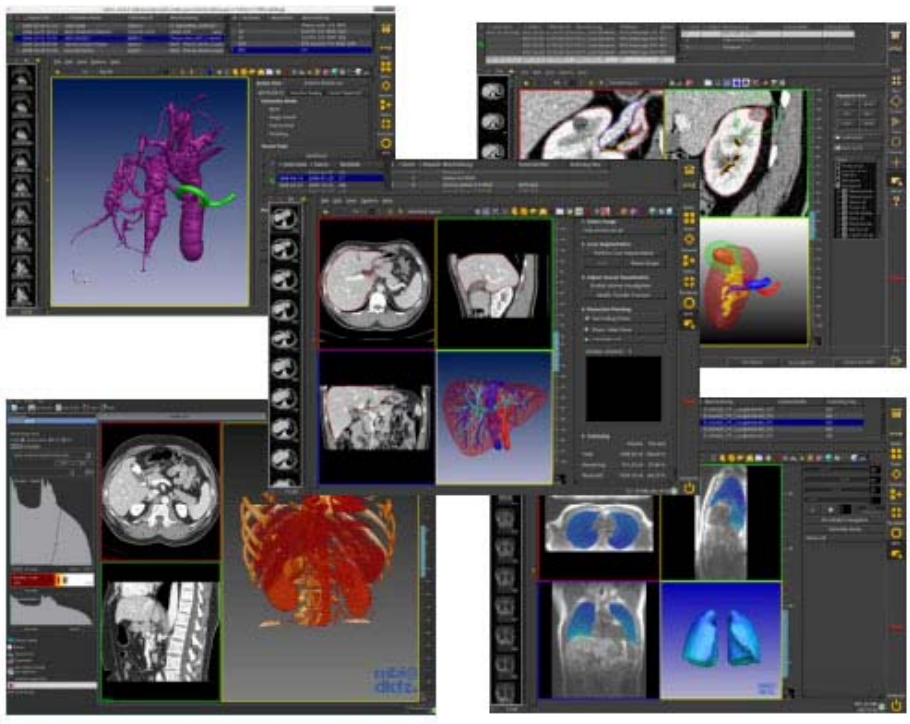


Abbildung 2.8: MITK Anwendungsbeispiele[Hei04]

Dabei besteht alleine der Open Source Teil aus ca. 500.000 Zeilen Code und ist etabliert in Forschung und Bildung mit ca. 200 aktiven Anwendern und Entwicklern (MIT Boston, RWTH Aachen, Charité Berlin usw.). Durch ein SVN Versionsverwaltung, das Bugtrackingsystem Bugzilla und Doxygen für die automatische Generierung der Dokumentation kann die Konsistenz und Qualität der Software sichergestellt werden. Zusätzlich gibt es die Möglichkeit, die Mailingliste für direktere Fragen zu nutzen und ein sogenanntes Dashboard einzusehen, um funktionierende Build Konfigurationen zu vergleichen.

Das MITK kann dafür verwendet werden, neuen Applikation zu entwickeln, die aktuelle Open-Source Version zu erweitern oder einfach die existierende Anwendung zu nutzen. Für das *Graphical User Interface (GUI)* wird QT verwendet und mit dem C++ Framework BlueBerry werden Architekturstandards für die Entwicklung festgelegt.

Die medizinischen Algorithmen werden zum Teil vom *Insight Segmentation and Registration Toolkit (ITK)* bereitgestellt. Die Visualisierung wird durch das *Visualization Toolkit (VTK)* ermöglicht. Als letzter Baustein für eine klinische Anwendung fehlen also nur noch Interaktionen die neben weiteren Features durch das MITK zur Verfügung gestellt werden[WVW+05]:

- Konsistente Ansicht (transversal, koronal, sagittal, 3D) auf medizinische Datensätze
- Zentrale Datenverwaltung zur Synchronisation der Ansichten
- Unterstützung für 4D (3D+t) Datensätze
- Interaktionen zum Erstellen oder Verändern von Daten
- Undo/Redo Funktionalität für Interaktionen und einzelne Datenprozesse
- Komplexe Interaktionen durch Verwendung von Zustandsdefinitionen
- Erweiterungen durch optionale Module für Segmentierung und Registrierung

Mehr Informationen sind auf der offiziellen Homepage[5] von MITK verfügbar.

2.4.2 BlueBerry

Der Begriff BlueBerry beschreibt ein modulares C++ Framework, das auf dem OSGi Standard[6] basiert. Es wurde für das MITK konzipiert und ist noch heute ein essentieller Bestandteil davon (siehe Abbildung 2.9).

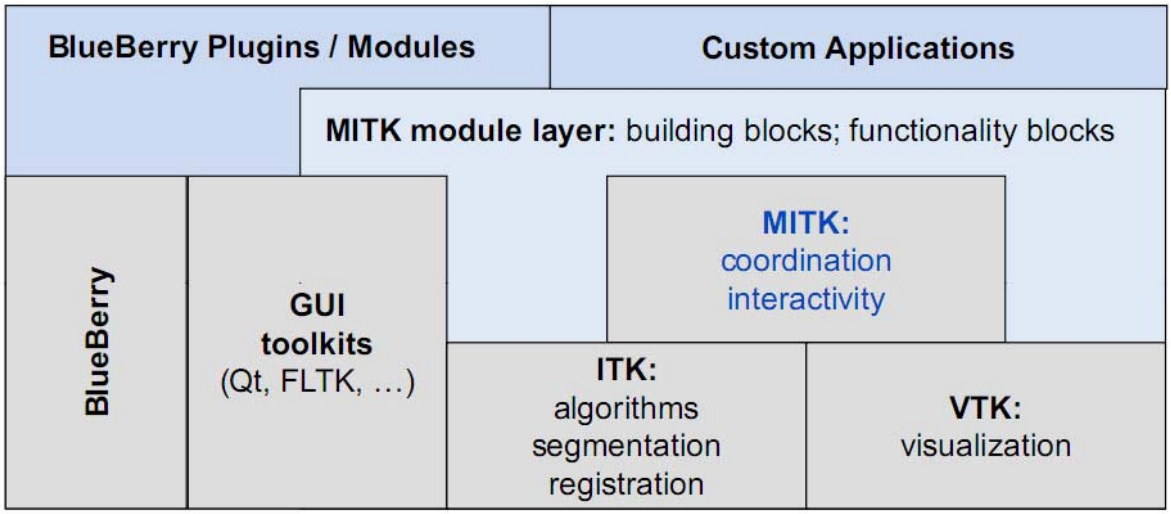


Abbildung 2.9: Überblick zu den Komponenten in MITK[MMS+10]

[5] http://www.mitk.org, Stand: 17-02-2011
[6] http://www.osgi.org/Main/HomePage, Stand: 17-02-2011

Das Framework verwendet zunächst Grundprinzipien der OSGi Spezifikation, wie das Benutzen von *Bundles* und einer *Service Oriented Architecture (SOA)*. Diese werden kombiniert mit dem Konzept der *Extension Points* aus Eclipse und ermöglichen damit ein dynamisches Verhalten zur Laufzeit[Mül10].

Service Oriented Architecture (SOA)

Ein Bundle im MITK Kontext beinhaltet Source Code, Deskriptoren für die verwendeten und bereitgestellten Services und deren Abhängigkeiten. Das Manifest innerhalb eines Bundles dient als Beschreibung über die Inhalte. Für jedes Bundle kann ein Activator, der eine Klasse oder ein Interface sein kann, zur Ausführung festgelegt werden. Dort wird der Startzustand definiert und alle nötigen Services geladen.

Eine SOA umfasst einen Service Provider, eine Registry und einen Service Consumer (siehe Abbildung 2.10).

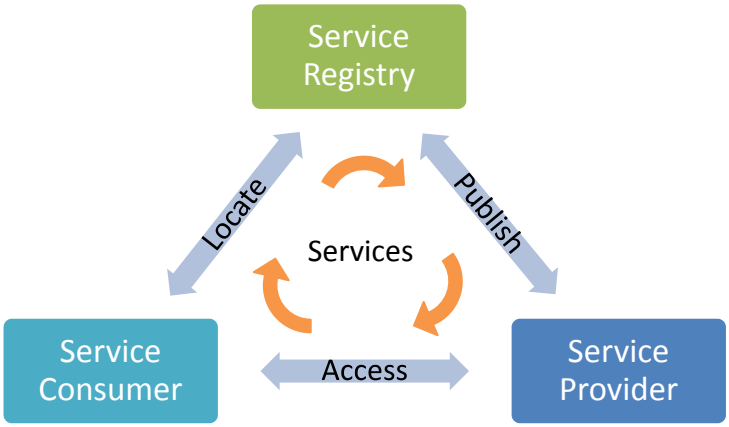


Abbildung 2.10: Überblick zur Service Oriented Architecture[Mül10]

Der Service Provider stellt sich selbst zur Verfügung, so dass er in die Service Registry eingetragen wird. Dort können alle angemeldeten Services vom Service Consumer über eine Suchanfrage gefunden. Dieser stellt dann eine direkte Zugriffsanfrage an den zugehörigen Service Provider, um den Service nutzen zu können.

Extension Points

In Eclipse hat sich das Konzept der Extension Points (siehe Abbildung 2.11) zur Verwaltung der Abhängigkeiten von Plugins als guter Standard herausgestellt.

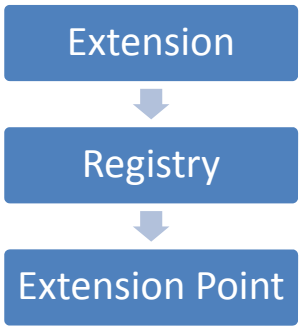


Abbildung 2.11: Überblick zum Konzept der Extension Points

Beim Start der Applikation werden alle Extensions in die Registry eingelesen. Extensions stellen Funktionalität zur Verfügung und werden mit Hilfe von XML Dateien definiert. Die Informationen über Extensions werden durch Deskriptoren in der Registry gespeichert und erlauben damit ein Aufrufen der benötigten Dateien nur bei Bedarf (Lazy Loading).

2.4.3 Qt

Die C++ Klassenbibliothek Qt (siehe Abbildung 2.12) wurde für die plattformunabhängige Realisierung von GUIs erstmals 1996 veröffentlicht. Sie ist Open Source, besitzt aber zusätzlich eine Lizenz für den kommerziellen Gebrauch und Support[Neu10].

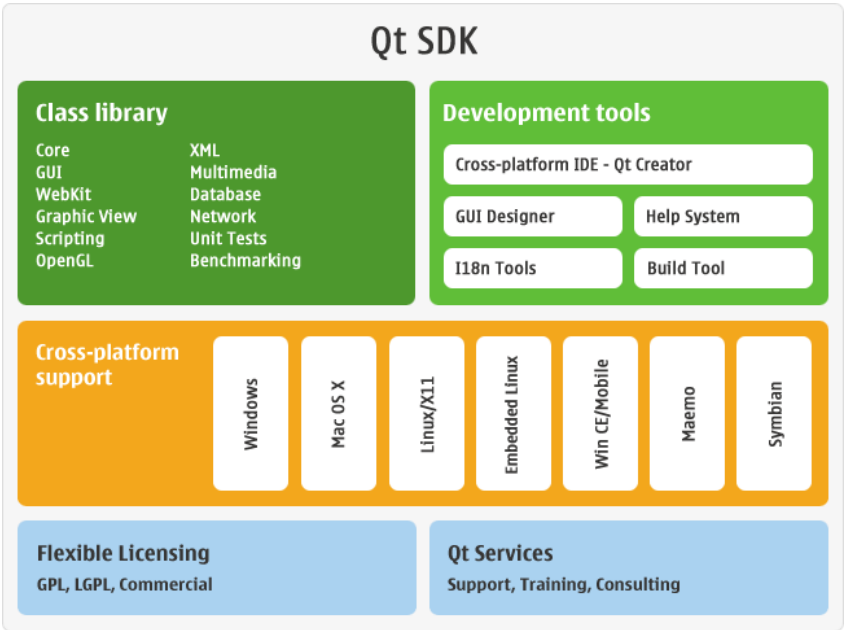


Abbildung 2.12: Überblick zu Qt[Neu10]

Der Core Bereich bietet Unterkategorien wie das Verwalten von Threads mit der Unterstützung für asynchrone Events. Dadurch können threadübergreifend Funktionen aufgerufen und Parameter übergeben werden. So kann ein Prozess einen anderen Prozess die Anweisung zur Ausführung einer Methode geben und gleichzeitig seine Eingabeparameter mit senden.

Der Core Bereich selbst ist nur einer von vielen Bereichen.In Abbildung 2.13 wird ein Überblick zu den am häufigsten verwendeten Komponenten dieser Bibliothek gegeben.

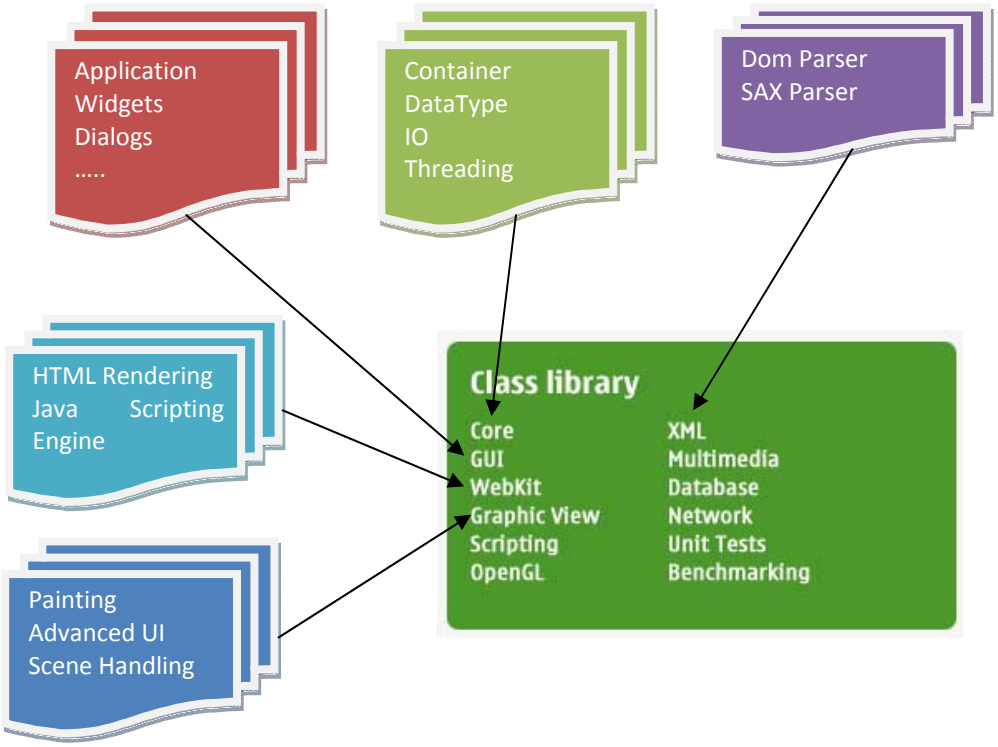


Abbildung 2.13: Detaillierte Betrachtung von Qt

Für detaillierte Informationen kann die offizielle Dokumentation von QT aufgerufen werden[QT10].

2.4.4 ITK

Nach einer Ausschreibung für ein Open Source Toolkit für Registrierung und Segmentierung der US National Library of Medicine und der National Insitutes of Health von 1999 wurde nach langer Entwicklungszeit erstmals 2002 das erste Release vom *Insight Segmentation and Registration Toolkit(ITK)* veröffentlicht. Die Wartung und Weiterentwicklung übernimmt das Insight Software Konsortium mit 6 Mitgliedern. Davon kommen 3 Partner aus der Industrie (GE, Kitware, Insightful) und 3 Partner aus der Forschung (University of North Carolina, University of Tennessee, University of Pennsylvania)[ISN+05]. Das Hauptziel von ITK ist die Bereitstellung von Datenstrukturen und Algorithmen für die Registrierung und Segmentierung

in medizinischen Anwendungen[MMS+10]. Dafür hält sich das Toolkit an grundlegende Softwareparadigmen. Zu diesen gehören generische Programmierung, Smart Pointer zur effizienten Speicherverwaltung, Event management, Multithreading etc. Außerdem nutzt es die VNL Bibliothek für mathematische Operationen[ISN+05].

Weitere Informationen sind auf der offiziellen Homepage[7] und in dem ITK Software Guide[ISN+05] verfügbar.

2.5 Mixed Reality

Mit *Mixed Reality* wird ein Oberbegriff für die zwei Begriffe *Virtual Reality (VR)* und *Augmented Reality (AR)* definiert. Eine VR versucht die Realität detailgetreu in einer computergenerierten Welt darzustellen (siehe Abbildung 2.15). Dagegen erweitert eine AR die Ansicht auf die Realität durch zusätzliche Informationen[Sch06](siehe Abbildung 2.14).

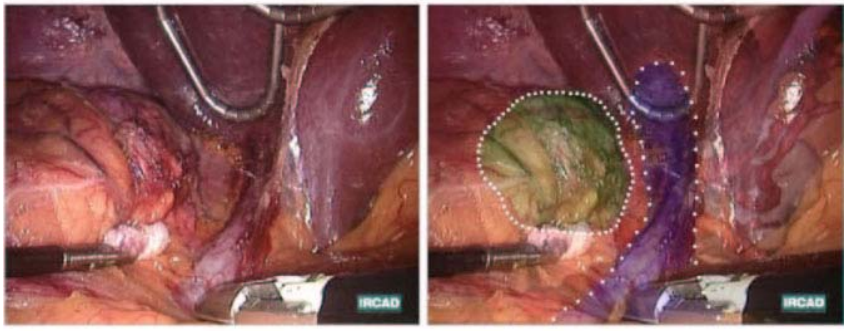


Abbildung 2.14: Augmented Reality in der Medizin[SNS+04]

Abbildung 2.15: Virtual Reality in der Medizin[FLW09]

Diese beiden Verfahren begegnen uns unbewusst schon überall im Alltag und drängen immer mehr in die klinische Applikation[SMR+09][SNS+04][FLW09]. Dabei versuchen sie Verfahren zu vereinfachen und zu verbessern, damit ein optimierter Behandlungsprozess gewährleistet werden kann.

[7] http://itk.org, Stand: 17-02-2011

2.6 Das Quaternion

Die Durchführung von Rotationen in einem dreidimensionalen Raum erfordert meistens eine Reihe von komplexen Rechenoperationen. Sir William Rowan Hamilton hat 1843 auf der Suche nach einer dreidimensionalen Erweiterung für komplexe Zahlen Quaternionen erfunden. In ihrem damaligen Zustand war es nur schwer möglich mit ihnen zu rechnen. Jedoch machte Josiah Willard Gibbs die Entdeckung, dass Quaternionen durch einen skalaren und vektoriellen Teil dargestellt werden können.

Dieser Fortschritt ermöglichte eine Vereinfachung für jegliche arithmetische Operationen mit Quaternionen. Seither werden sie verstärkt im Zusammenhang mit Rotationen von 3D Objekten eingesetzt[Bar01]. Die Verwendung von Quaternionen gegenüber Rotationsmatrizen bietet mehrere Vorteile. Wegen ihrer geringen Größe verbrauchen sie erheblich weniger Speicherplatz. Außerdem sind, wie bereits erwähnt, Rechenoperationen leichter realisierbar, weil bei einer Konkatenation (Ausführung mehrerer Abbildungen hintereinander) von zwei Quaternionen weniger Rechenschritte benötigt werden[Len04].

Definition der Grundform

Grundsätzlich gibt es zwei verschiedene Schreibweisen für Quaternionen. Die erste ist eine Darstellung im komplexen Zahlenraum (siehe Formel 2.3) und die zweite eine eher allgemeinere Darstellung (siehe Formel 2.4). Es gibt noch weitere Formen (Polarkoordinaten, Rotationsmatrizen), die je nach Anwendungsfall verschieden definiert werden.

Die Imaginärteile werden mit i, j, k bezeichnet und die jeweils darauf folgenden Variablen x, y, z stellen den vektoriellen Anteil dar. Gleichzeitig bilden x, y, z den Vektor mit dem die Rotationsachse beschrieben werden kann. Der reelle Wert s ist der Skalarteil eines Quaternion.

$$q = s + ix + jy + kz$$

Formel 2.3: Grundform eines Quaternion im komplexen Zahlenraum

$$q = \left[s, \begin{pmatrix} x \\ y \\ z \end{pmatrix}\right]$$

Formel 2.4: Allgemeine Darstellung eines Quaternion

Rechenregeln für Imaginärteile

Für die Imaginärteile i, j, k gelten hier besondere Rechenregeln (siehe Formel 2.5), die denen von komplexen Zahlen ähneln. Bei der Multiplikation muss allerdings darauf geachtet werden, dass diese für Quaternions nicht kommutativ sind.

$$i^2 = j^2 = k^2 = -1$$

$$i * j = k$$

$$j * k = i$$

$$k * i = j$$

$$j * i = -k$$

Formel 2.5: Rechenregeln für Imaginärteile von Quaternions

Betrag (Länge) eines Quaternions

Die Länge $\|q\|$ eines Quaternions ist in Formel 2.6 definiert und für ein Einheitsquaternion stets 1. Diese Sonderform von Quaternionen erlauben eine Vereinfachung von Rechnungen und werden deshalb oft verwendet.

$$\|q\| = \sqrt{s^2 + x^2 + y^2 + z^2}$$

Formel 2.6: Definiton des Betrags (der Länge) von Quaternions

Multiplikation von Quaternionen

Eine für die Rotation relevante Rechenregel ist die Multiplikation von Quaternionen. Durch einen Beweis mit Quaternion q_1 und q_2 (siehe Formel 2.7) soll die allgemeine Rechenregel für das Multiplizieren hergeleitet werden. Der skalare (vektorielle) Anteil wird dabei mit s_i (v_i) beschriftet und der Index i definiert die Zugehörigkeit zu einem Quaternion.

$$\begin{aligned} q_1 * q_2 &= [s_1, v_1] * [s_2, v_2] \\ &= (s_1 + ix_1 + jy_1 + kz_1) * (s_2 + ix_2 + jy_2 + kz_2) \\ &= s_1 * s_2 - (x_1x_2 + y_1y_2 + z_1z_2) \\ &+ i * (s_1x_2 + s_2x_1 + y_1z_2 - z_1y_2) \\ &+ j * (s_1y_2 + s_2y_1 + z_1x_2 - x_1z_2) \\ &+ k * (s_1z_2 + s_2z_1 + x_1y_2 - y_1x_2) \\ &= [s_1s_2 - v_1v_2, v_1 \times v_2 + s_1v_2 + s_2v_1] \end{aligned}$$

Formel 2.7: Herleitung der Multiplikation von Quaternions

Darstellung in Polarkoordinaten

Eine Verwendung von Polarkoordinaten[Kor86] ermöglicht es ein Einheitsquaternion durch einen Winkel θ und eine entsprechende Rotationsachse $\vec{r} = \{x, y, z\}$ darzustellen (siehe Formel 2.8).

$$q = \cos\frac{\theta}{2} + \sin\frac{\theta}{2} * (ix + jy + kz)$$

Formel 2.8: Darstellung eines Einheitsquaternion mit Polarkoordinaten

Überführung in eine Rotationsmatrix

Prinzipiell kann aus Einheitsquaternion q eine Rotationsmatrix R gemacht werden[Kor10]. Dabei wird R durch die Formel 2.9 aus dem Quaternion q erstellt.

$$q = q_0 + iq_1 + jq_2 + kq_3$$

$$R = \begin{pmatrix} (q_0^2 + q_1^2 - q_2^2 + q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_2q_1 + q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_3q_1 - q_0q_2) & 2(q_3q_2 + q_0q_3) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{pmatrix}$$

Formel 2.9: Darstellung eines Einheitsquaternion in einer Rotationsmatrix

2.7 Posentransformation

Eine Pose P wird durch Position (Translation) und Orientierung (Rotation) charakterisiert. Sie beschreibt die Position und Orientierung eines Objektes relativ zu einem Bezugskoordinatensystem. Die Pfeilrichtung in einer Abbildung gibt den Bezugspunkt für die Pose an. Um von einer Pose P_i zur nächsten Pose P_{i+1} zu gelangen, werden Posentransformationen ΔT_{i+1} verwendet (siehe Abbildung 2.16). Diese Transformationen definieren welche Änderungen notwendig sind, damit die neue Pose erreicht werden kann. In Abbildungen gibt ein Pfeil die Richtung einer Transformation an. Jedes mit einer Pose dargestellte Objekt besitzt ein eigenes lokales Koordinatensystem[Wol07].

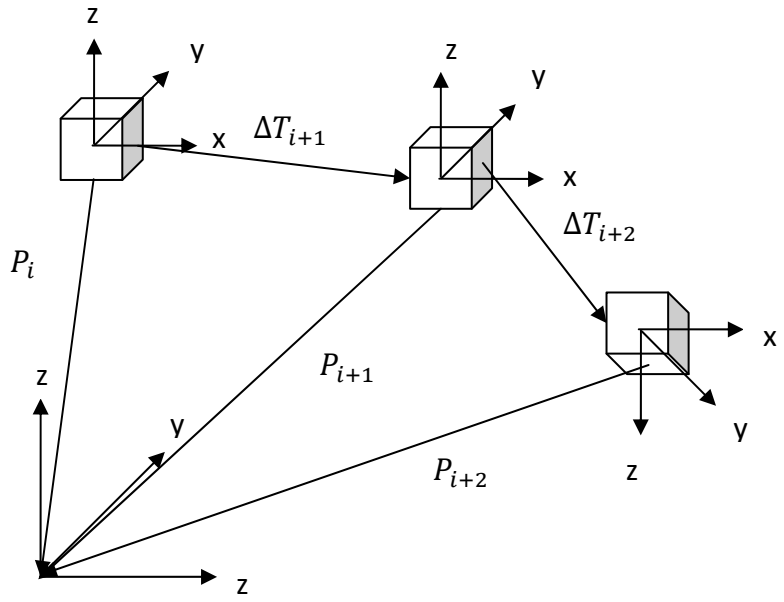


Abbildung 2.16: Beispiel für eine Reihe von Posentransformationen

Für die Berechnung einer Transformation ist es notwendig, dass die Richtung geändert werden kann. Dieser Vorgang wird durch das Verwenden einer inversen Transformation ermöglicht. Außerdem wird eine Transformation von einer Pose eines Objektes zu einer Pose eines anderen Objektes durch $T_{Von\ Objekt}^{Nach\ Objekt}$ dargestellt (siehe Abbildung 2.17).

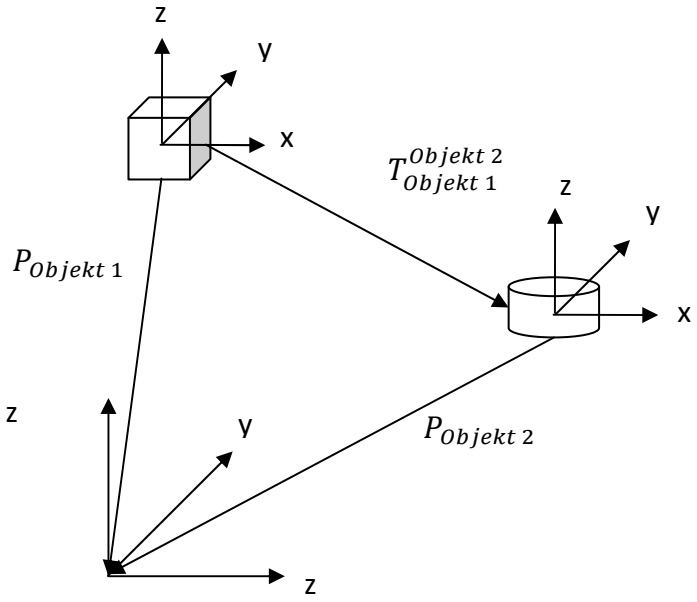


Abbildung 2.17: Beispiel für Posentransformation zwischen unterschiedlichen Objekten

Damit eine Matrixmultiplikation durchgeführt werden kann, sind sowohl die Pose P, als auch die Posentransformation T durch eine 4x4 Matrix mit homogenen Koordinaten dargestellt (siehe Formel 2.10). Diese Matrix besitzt Rotationskomponenten R (Orientierung) mit entsprechenden Indizes und die Translationskomponenten T_x, T_y, T_z (Position).

$$P,T = \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Formel 2.10: Pose und Posentransformation als Matrix

Anhand der in Abbildung 2.17 gezeigten Transformation wird die typische Schreibweise einer Matrixmultiplikation vorgestellt (siehe Formel 2.11). Dabei wird stets die zuerst ausgeführte Transformation auf die rechte Seite geschrieben.

$$P_{Objekt\ 1} = P_{Objekt\ 2} * T^{Objekt\ 2}_{Objekt\ 1}$$

Formel 2.11: Definition der Schreibweise für Matrixmultiplikationen mit Posen

2.8 Stand der Technik

Die Medizintechnik bietet eine große Bandbreite von teuren Trackingsystemen basierend auf den unterschiedlichsten Trackingverfahren. Kapitel 2.8.1 soll dabei einen Überblick über die in der Forschung verwendete Trackingsysteme geben. Jedoch lassen sich bereits mit low-cost Trackingssystemen wie der Wiimote intuitive Interaktionskonzepte entwickeln. Erste Beispiele werden in Kapitel 2.8.2 aufgelistet. Außerdem werden in dieser Arbeit weitere Konzepte mit der Wiimote entwickelt, die vor allem für den Gebrauch in der Medizin von Nutzen sind.

2.8.1 Tracking Systeme

Im Folgenden wird zwischen den zwei meist genutzten Trackingverfahren unterschieden. Dem optischen und elektromagnetischen Tracking. Beide besitzen ihre Vor- und Nachteile und werden je nach Anwendungsbereich passend ausgewählt. So kann bei einem elektromagnetischen Tracking durch metallische Gegenständen Interferenzen erzeugt werden. Dafür spielen Lichteinflüsse und optische Hindernisse keine Rolle - im Gegensatz zum optischen Tracking. Die hier vorgestellten Trackingsysteme haben je nach Konfiguration eine Preisspanne von ca. 8000 bis 15000€.

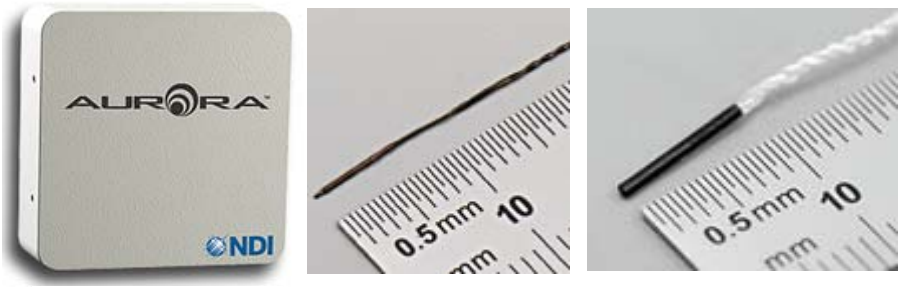
Die Firma Northern Digital Inc. (NDI) hat sich unter anderem darauf spezialisiert, Trackingsysteme für den klinischen Gebrauch zu produzieren. Es gibt das Polaris Spectra oder Vicra (siehe Abbildung 2.18). Beide bieten neben einer hohen Präzison (Abweichungen im Bereich 0.25-0.35mm) zusätzliche Features wie kurze Aufwärmzeit, einfache Integration durch ein geeignetes Programmierinterface, Robustheit gegenüber Repositionierungen usw[8].

Abbildung 2.18: Polaris Spectra und Polaris Vicra der Firma NDI[9]

Im Bereich elektromagnetisches Tracking stellt NDI das Aurora (siehe Abbildung 2.19) zur Verfügung. Es bietet Features von der einfachen Integration durch gut dokumentierte Programmiererinterfaces über 5DoF und 6DoF Sensoren für in-vivo Applikationen bis hin zu

[8] http://www.ndigital.com/medical/polarisfamily-benefits.php, Stand: 19-02-2011
[9] http://www.ndigital.com/medical/polarisfamily.php, Stand: 19-02-2011

Kompensation von typischen metallischen Störquellen in der Medizin[10]. Bei Messungen mit beispielsweise dem 5DoF Tool ist von Abweichungen zwischen 0.7 und 0.9mm für die Position und zwischen 0.1 bis 0.3 Grad für die Orientierung zu rechnen[11].



(a) Aurora (b) 5DoF Tool (c) 6DoF Tool

Abbildung 2.19: Aurora von NDI mit 5DoF und 6DoF Sensoren[12]

Ein weiteres optisches Trackingsystem ist der Micron Tracker von Clarontech (siehe Abbildung 2.20). Je nach Modell kann es zur Kalibrierungsabweichungen von 0.20 bis 0.35mm kommen. Bei Messungen mit einem statischen/bewegenden Objekt sind es 0.007/0.07 bis 0.015/0.14 mm[13]. Weiterhin gehört der Micron Tracker zur dritten Generation von optischen Trackern, die vollkommen passiv tracken können. Dabei erkennen sie starke Kontraste im Lichtspektrum und können diese als Muster erkennen[14].

Abbildung 2.20: Micron Tracker von Clarontech[15]

[10]http://www.ndigital.com/medical/aurora-benefits.php, Stand: 19-02-2011
[11]http://www.ndigital.com/medical/aurora-techspecs-performance.php, Stand: 19-02-2011
[12]http://www.ndigital.com/medical/aurora.php und http://www.ndigital.com/medical/aurora-sensors.php, Stand: 19-02-2011
[13]http://www.clarontech.com/microntracker_products.php, Stand: 19-02-2011
[14]http://www.clarontech.com/microntracker_technology.php, Stand: 19-02-2011
[15]http://www.clarontech.com/platforms_microntracker.php, Stand: 19-02-2011

2.8.2 Wiimote-Projekte

Die folgenden Projekte sind Beispiele für die Entwicklungsmöglichkeiten der Wiimote in der Wissenschaft:

Das erste Projekt war das Toolkit TaKG zur Gestenklassifikation[Neß08]. Damit sollten Benutzereingaben durch die Verwendung von Gesten vereinfacht werden. Durch die integrierte Kamera (siehe Kapitel 2.3.2) der Wiimote konnten Merkmalextraktionen sowie das Erlernen und Wiedererkennen von Gesten realisiert werden. Die durch eine Evaluation festgelegte Präzision der erkannten Gesten lag bei 72%.

Durch die Kombination von zwei Wiimotes war es möglich, ein Stereokamerasystem zu konstruieren[HNH08]. In einem festgelegten Volumen können IR Sender in Echtzeit mit einer Standardabweichung von 2.23mm getrackt werden. So konnte durch einen low-cost Versuch bereits ein komplexes Stereokamerasystem zum Prototyping entwickelt werden.

Auch im medizinischen Sektor hat der Einsatz von Eingabegeräte Anwendung gefunden. Im Zusammenhang mit Ultraschall gestützter Anästhesie wurde eine VR Applikation mit den inertialen Sensoren umgesetzt[SLH+10]. Die Wiimote wurde als Ultraschallsonde und Injektionsnadel in eine virtuelle Umgebung abgebildet und zur Durchführung einer Übung getrackt. Dadurch wurde bewiesen, dass sich die Verwendung von low-cost Hardware auch für Trainingssimulationen in der Medizin auszahlt.

3 Ergebnisse

Im Rahmen dieser Arbeit wurde ein Design zur Integration von externen Eingabegeräten basierend auf BlueBerry (siehe Kapitel 2.4.2) erarbeitet und anhand von zwei Anwendungsbeispielen implementiert. Ziel war es zu zeigen, dass durch intuitive Interaktionen mit low-cost Eingabegeräten die Kommunikation zwischen Benutzer und Computer im klinischen Behandlungsprozess vereinfacht werden kann. Dabei spielt die Genauigkeit der Anwendungsspiele eine untegeordnete Rolle, da eine benutzerfreundliche und intuitive Interaktion immer von der subjektiven Perspektive abhängig ist.

In Kapitel 3.1 wird auf die entwickelte Architektur des Designs und die Integration der Wiimote (Game controller, Nintendo, Japan) und des Space Navigators (3D Maus, Analog Devices, USA) ins MITK eingegangen. Dabei wurde ein Open-Source Treiber (siehe Kapitel 2.3.5) verwendet und eigene Klassen zur Verarbeitung von Sensordaten, zum Erstellen von eigenen Funktionalitäten und für die Definition von Kommunikationsschnittstellen implementiert.

Anschließend wird in Kapitel 3.2 ein virtuelles Headtracking durch die Infrarotkamera (siehe Kapitel 2.3.2) mit einem transversalen Scrollen der Schichtbilder näher erläutert. Dafür wurden ein Controller zum Interagieren mit den medizinischen Daten und ein eigener Kalibrierungsprozess auf Basis der Zustandsautomaten in MITK entwickelt. Mit den Universitätsklinikum Heidelberg wurde ein Proof of Concept für die Anwendung des VR Head-trackings ausgearbeitet und durchgeführt.

Die Interaktion unter Verwendung von 6 Freiheitsgraden (siehe Kapitel 2.3.1) im dreidimensionalen Raum befindet sich in Kapitel 3.3. Für die Darstellung der vollständigen Pose aus Orientierung und Position wurden die Sensoren der Wiimote genutzt (siehe Kapitel 2.3.3 und 2.3.4). Die Änderung der Orientierung (Rotation) wurde durch Quaternionen (siehe Kapitel 2.6) der VNL Bibliothek (siehe Kapitel 2.4.4) realisiert. Für die Translation wurden mit Hilfe physikalischer Grundgesetze[TW04] eine Kompensation der Schwerkraft entwickelt und implementiert. Die zwei entwickelten Modi für eine Posentransformation (siehe Kapitel 2.7) von 3D Objekten wurden durch Matrizenmultiplikation in einem eigenen Interactor (Steuerungsklasse) implementiert.

3.1 Externe Eingabegeräte

Damit externe Eingabegeräte verwendet werden können, müssen die zugehörigen Treiber in die Anwendung eingebunden werden. Da dieser Integrationsprozess wegen unterschiedlich aufgebauten Treibern variiert, war die Entwicklung von standardisierten Interfaces und einem geeignetem Konzept notwendig. Es wurde von BlueBerry (siehe Kapitel 2.4.2) die Service Oriented Architecture und das Konzept der Extension Points aufgegriffen und für eine Anwendung mit externen Eingabegeräten angepasst. Dadurch soll Modularität und einfache Wiederverwendbarkeit der implementierten Klassen gewährleistet werden. Für die Treiberkommunikation wurde für das Multithreading Qt (siehe Kapitel 2.4.3) und zur Weitergabe von Informationen ITK (siehe Kapitel 2.4.4) verwendet.

3.1.1 Modellierung

Dieser Abschnitt beschreibt den Aufbau der entwickelten Integrationsstruktur. Zur Darstellung der Zusammenhänge zwischen den einzelnen Komponenten wird ein kurzer Überblick (siehe Abbildung 3.1) gegeben.

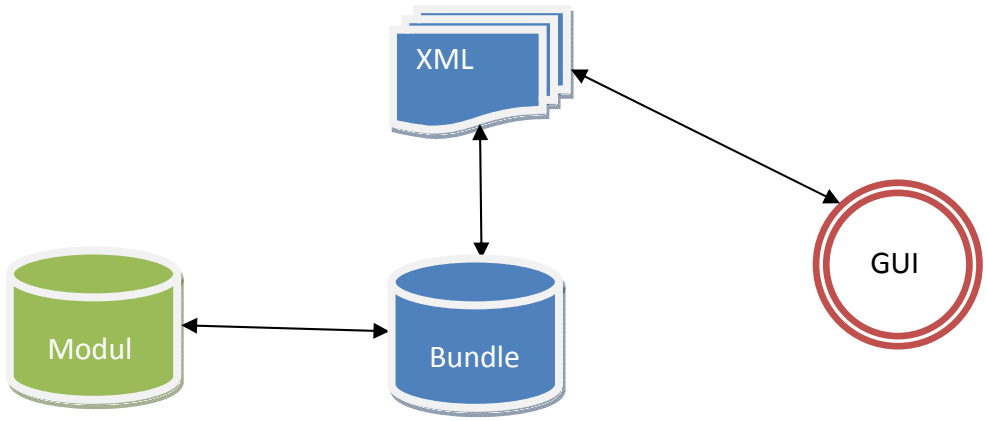


Abbildung 3.1: Überblick Integrationsstruktur

Ein *Modul* ist ein Baustein in MITK (siehe Abbildung 3.1), der einer Hauptapplikation zusätzliche Funktionalität zur Verfügung stellt. Im Integrationsprozess wird es dazu verwendet, die Treiberklassen für Eingabegeräte und alle nötigen Schnittstellen und Funktionen zu verwalten. Über das Buildsystem *CMake* können Module vor der Laufzeit der Anwendung hinzugefügt werden.

Damit ein dynamisches Aktivieren sowie Deaktivieren von Eingabegeräten während der Laufzeit möglich ist, wurden *Bundle* (siehe Kapitel 3.1.3) vom BlueBerry Framework (siehe Kapitel 2.4.2) eingesetzt. Diese erlauben mit Hilfe von *Extension Points* (siehe Kapitel 3.1.3) den Zugriff auf XML Dateien während des aktiven Betriebs des Programms.

3.1.2 Modul

Für eine genauere Betrachtung des Moduls wird ein Klassendiagramm (siehe Abbildung 3.2) des für die Wiimote entwickelten Moduls aufgeführt und anhand dessen die verwendete Architektur erklärt. Für eine detaillierte Klassenauflistung wird auf die MITK Dokumentation[16] verwiesen.

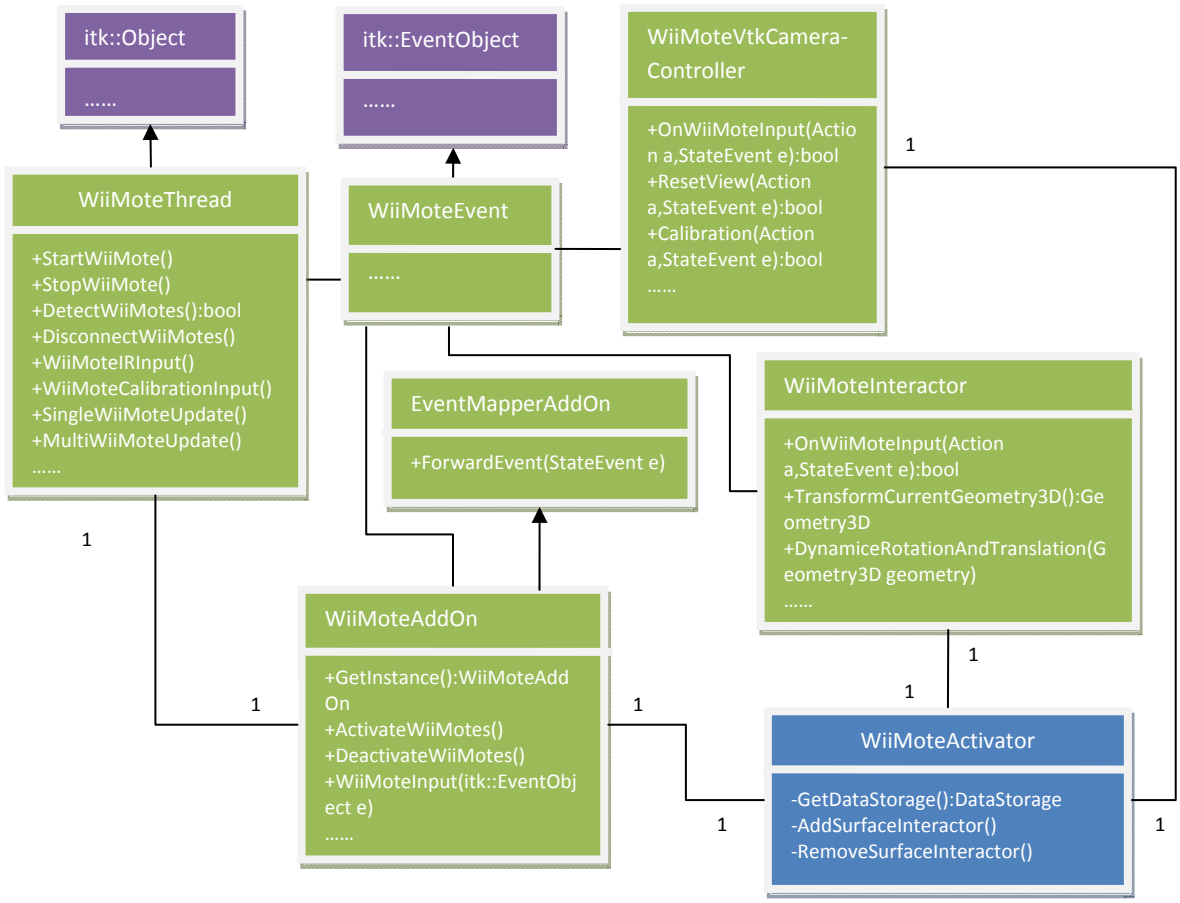


Abbildung 3.2: Klassendiagramm WiiMote

Zunächst wird die Threadklasse (siehe in Abbildung 3.2 WiiMoteThread) betrachtet. Dort werden zwei Hauptaufgaben durchgeführt: die direkte Kommunikation mit dem Treiber und die Weiterverarbeitung der ankommenden Daten (siehe Abbildung 3.3). Durch die Verwendung von ITK-Prozessen wird einerseits dafür gesorgt, dass die laufende Anwendung nicht blockiert wird. Andererseits können rechenintensive Operationen parallel durchgeführt werden.

Nachdem die Informationen erfolgreich verarbeitet wurden, werden diese in entsprechender Form (siehe in Abbildung 3.2 WiiMoteEvent) weitergegeben werden. Je nach Anforderung wird dafür ein eigener Eventtyp definiert. Dieser muss von *itk::EventObject* erben, da sonst ein Transport von Thread zu Thread nicht gewährleistet ist. Für diesen Vorgang wird

[16] http://www.mitk.org/wiki/Documentation, Stand: 25-01-2011

die *Callbackfunktion* von QT (siehe Kapitel 2.4.3) verwendet. Solche Funktionen erlauben die Übergabe von Funktionen als Parameter. Damit kann ein Prozess einem anderen Prozess die Ausführung einer Methode befehlen und zusätzlich in der übergebenen Funktion Daten als Parameter übergeben (siehe Abbildung 3.3).

Sind die Daten erfolgreich in der Hauptanwendung beim passenden *AddOn* (siehe in Abbildung 3.2 WiiMoteAddOn) angekommen, können diese Events auf die zugehörigen Listener weitergeleitet werden (siehe Abbildung 3.3). Mit einem AddOn wird die Kommunikationsschnittstelle in Form einer Eventverwaltung zwischen Treiber und dem *EventMapper* bezeichnet. Der EventMapper sorgt dafür, dass jedes Event einer eindeutigen ID zugeordnet ist und übergibt diese Information an die globale Interaktionsverwaltung (siehe Abbildung 3.3). Deswegen muss jedes AddOn auch von der Schnittstelle *mitk::EventMapperAddOn* erben, damit es vom EventMapper als gültiges Objekt anerkannt wird.

Innerhalb des Controllers oder der Interactors (siehe Abbildung 3.2 WiiMoteVtkCameraController und WiiMoteInteractor) werden durch Events vordefinierte Aktionen durchgeführt. Anhand des folgenden Sequenzdiagrams (siehe Abbildung 3.3) wird der in diesem Kapitel beschriebene Ablauf innerhalb des Systems verdeutlicht. Als Beispielablauf wird die Eingabe eines IR Signals durch den Benutzer beim VR Headtracking (siehe Kapitel 3.2) verwendet.

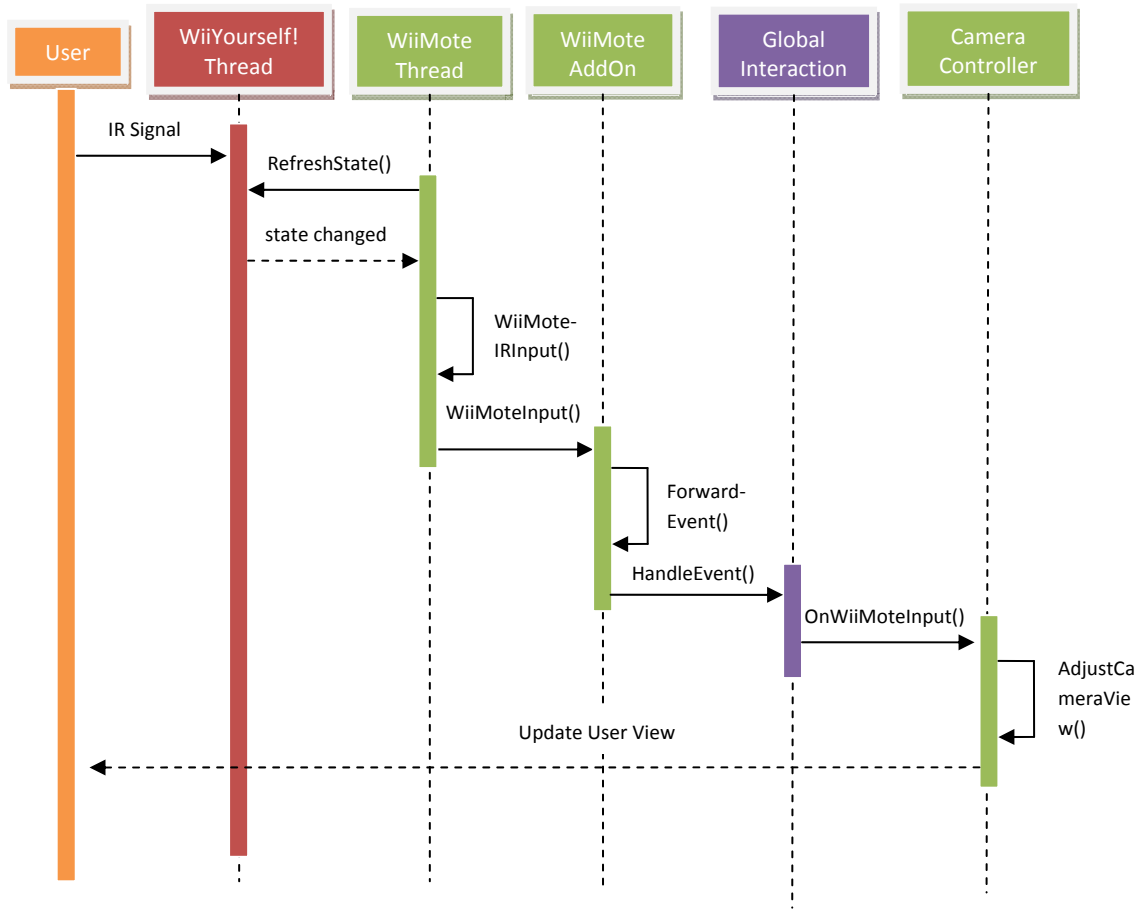


Abbildung 3.3: Sequenzdiagramm WiiMote Headtracking

3.1.3 Bundle

Im Folgenden soll die Verwendung von Bundles und Extension Points im Zusammenhang mit dem Integrationsprozess durch ein Klassendiagramm erklärt werden (siehe Abbildung 3.4). Die Präfixe „I“ vor Klassen stehen als Abkürzung für „Interfaces“.

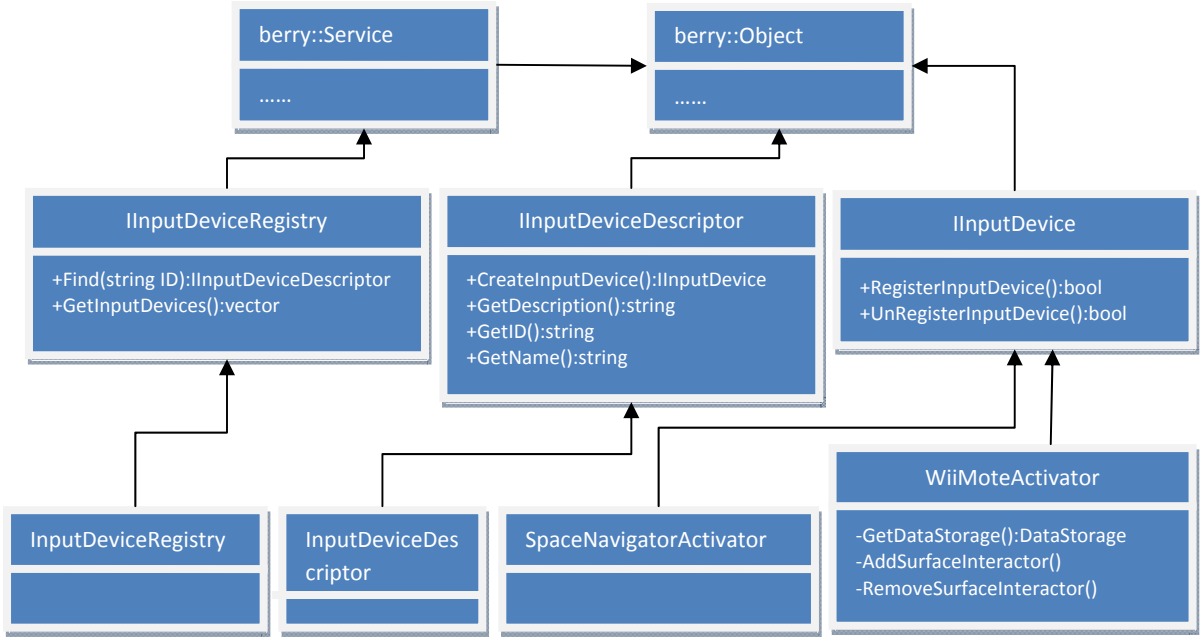


Abbildung 3.4: Klassendiagramm Bundle

Über die Verbindung zu BlueBerry (siehe Kapitel 2.4.2) werden Extension Points verwendet. Das ist ein Konzept, das über XML Dateien und Interfaces dynamisches Verhalten während der Laufzeit ermöglicht. Im Bundle wird eine XML Datei erstellt, die eine Erweiterung zu einem Extension Point für Input Devices darstellt. Bei Aktivierung eines Bundles wird die Information in der XML Datei dazu genutzt das Eingabegerät dem Extension Point hinzuzufügen. Durch die Interfaces ist es möglich während der Laufzeit Methoden auszuführen, die je nach Implementierung einen anderen Startzustand für das jeweilige Eingabegerät bieten.

Im Folgenden wird auf die drei bereitgestellten Interfaces aus Kapitel 2.4.2 und deren Anpassung an diese Applikation genauer eingegangen:

- Die *Registry* verwaltet alle vorhandenen Eingabegeräte in Form von Deskriptoren. Beim Programmstart werden aus einer XML Datei die aktuellen Einstellungen ausgelesen und übernommen. Es kann dort nach bestimmten *Input Devices* (Eingabegeräten) gesucht werden und alle vorhandenen ausgegeben werden.
- Ein Eingabegerät wird durch einen *Deskriptor* beschrieben. Dieser enthält eine kurze Beschreibung, eine eindeutige Identifikationsnummer (ID) und einen zugewiesenen Namen. Außerdem kann er den direkten Zugriff auf ein Input Device ermöglichen.
- Ein *InputDevice* muss standardmäßig eine Methode zum Registrieren und Deregistrieren besitzen. Dort wird definiert welche Instanzen (AddOn, Thread, Controller/Interactor) erzeugt oder gestartet werden müssen. Des Weiteren werden diese an den richtigen Stellen (EventMapper, globale Interaktionsverwaltung) angemeldet.

3.1.4 3D Maus

Als erstes Beispiel soll der Space Navigator von Analog Devices (siehe Kapitel 2.2) integriert werden und eine Steuerung der virtuellen Kamera umgesetzt werden. Als Überblick dient das Klassendiagramm des Space Navigators (siehe Abbildung 3.5).

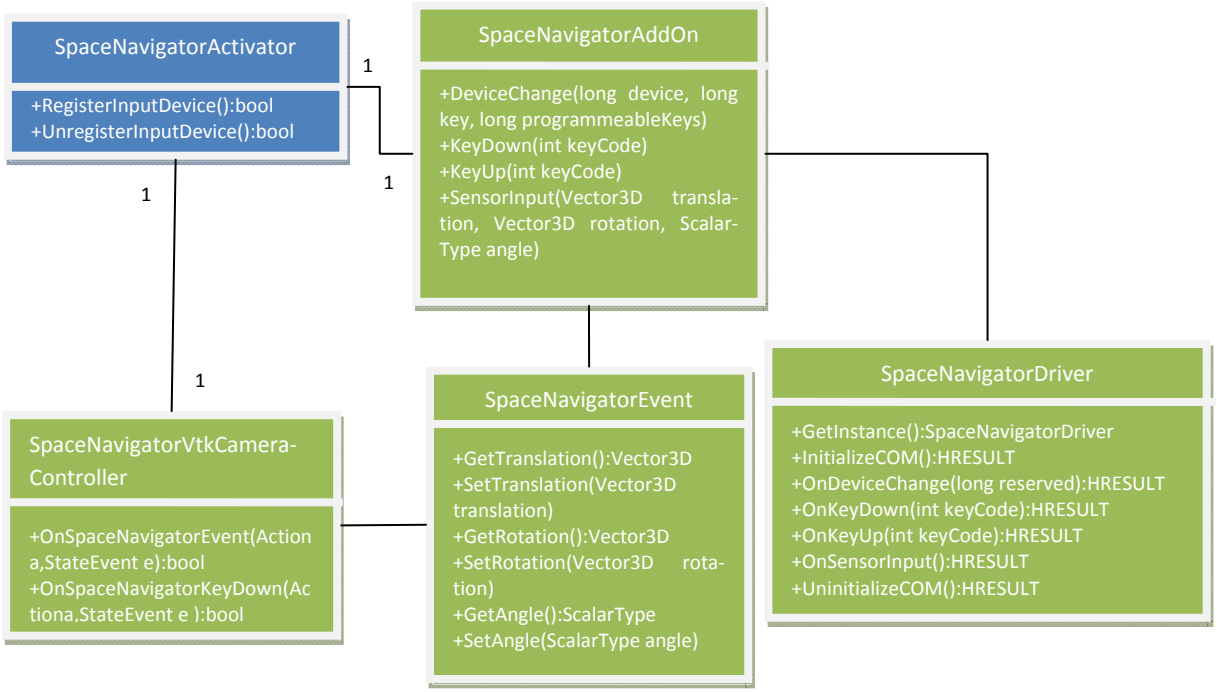


Abbildung 3.5: Klassendiagramm SpaceNavigator

Da die 3D Maus bereits eigene Treiber zur Verfügung stellt, war es nicht notwendig, einen eigenen Prozess für den *SpaceNavigatorDriver* zu verwenden. An dieser Stelle wird trotzdem über eine vom Treiber bereitgestellte Schnittstelle mit dem eigentlichen Treiber auf dem Betriebssystem kommuniziert. Wird die Maus bewegt oder ein Knopf gedrückt, so wird ein Event von der Treiberklasse registriert und nach Verarbeitung an das *SpaceNavigatorAddOn* weitergeleitet.

Durch die Aktivierung des Bundles kann das *SpaceNavigatorAddOn* die Events empfangen und an die entsprechenden globalen Instanzen weiterreichen. Erreicht ein Event den Controller, so können dort die Informationen in Aktionen umgewandelt werden. Eine Translation wird so als eine geradlinige Bewegung der Kamera entlang der entsprechenden Achse interpretiert. Ebenso eine Rotation als Drehung um eine passende Achse. Dadurch können komplexe Kamerafahrten mit Hilfe der 3D Maus (siehe Kapitel 2.2) durchgeführt werden. Zusätzlich wurde eine Resetfunktion eingebaut, sodass die Kameraperspektive auf eine Überblicksposition zurück gesetzt werden kann.

3.1.5 Open Source Treiber

Als zweites Integrationsbeispiel wurde die Wiimote von Nintendo (siehe Kapitel 2.3) ausgewählt. Damit eine Verwendung möglich ist, musste ein geeigneter Treiber ausgesucht werden. Die WiiYourself! Bibliothek erfüllt die gewünschten Anforderungen (C++, freie Lizenz, Support) und steht nur unter Windows zur Verfügung. Dieses Problem entsteht durch die Architektur der Bibliothek, die das *Windows Driver Kit (WDK)* und das Windows *Software Development Kit (SDK)* benutzt. Für eine korrekte Installation muss die Entwicklungsumgebung gemäß Abbildung 3.6 angepasst[17], werden (kann je nach Version des Betriebssystems variieren).

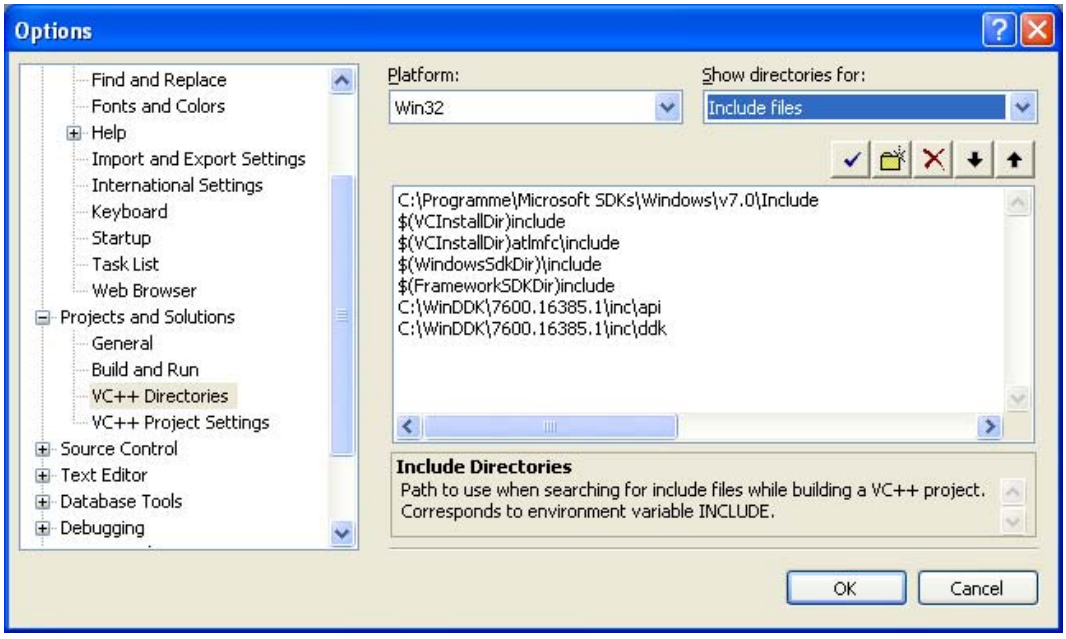


Abbildung 3.6: Windows Build Settings

Die Integration für die Wiimote wurde bereits in ausführlich in Kapitel 3.1.2 beschrieben. Weitere Details sind in der MITK Dokumentation[18] aufgeführt.

[17] http://social.msdn.microsoft.com/Forums/en-US/vcgeneral/thread/522a7180-c6aa-439f-963f-0ed10d49a239, Stand: 15-02-2011

[18] http://www.mitk.org/wiki/Documentation, Stand: 25-01-2011

3.2 VR Headtracking

Um medizinischen Daten wie Schichtbilder oder Operationsplanungen mit dreidimensionalen Volumenvisualisierungen während einer OP zu benutzen, müssen momentan stets Maus und Tastatur bedient werden. Der operierende Arzt muss dabei einer weiteren Person mitteilen, welches Bild gerade benötigt wird. Durch ein *VR Headtracking* wird diese Interaktion ohne eine zusätzliche Person ermöglicht.

Zu Beginn wird das Verfahren des IR Tracking mit der integrierten Kamera der Wiimote in Kapitel 3.2.1 beschrieben. Die entwickelten Klassen folgen den in MITK festgelegten Standards (Aufbau und Namenskonventionen) und verwenden standardisierte Mechanismen (Event – Listener) zur Weitergabe von Informationen. Zusätzlich wird in Kapitel 3.2.2 ein speziell entwickelter Kalibrierungsprozess mit den Zustandsmaschinen des MITK umgesetzt. Damit kann die Sensitivität für unterschiedliche Distanzen zwischen Benutzer und Kamera oder den persönlichen Präferenzen entsprechend eingestellt werden. Außerdem wird die bereits entwickelte Funktionalität um ein Scrollen durch transversale Schichtbilder mit einem IR Stift in Kapitel 3.2.3 erweitert. Abschließend wird in Kapitel 3.2.4 die Anwendung innerhalb eines OP Saals geschildert und auf weitere Probleme eingegangen.

3.2.1 IR Tracking

Der Begriff VR Headtracking setzt sich aus zwei Teilen zusammen. Eine virtuelle Realität (siehe Kapitel 2.5) definiert eine computergenerierte, dreidimensionale Welt, die versucht einen Sachverhalt aus der Realität detailgetreu abzubilden. Dieser Sachverhalt wird durch Headtracking näher eingegrenzt. Hier wird versucht, die reelle Kopfbewegung des Nutzers zu verfolgen. Dann werden diese Daten in das System übersetzt und dienen dafür, die optische Wahrnehmung auf ein 3D Objekt in der Wirklichkeit zu imitieren. Zur Realisierung eines solchen Konzepts werden die unterschiedlichsten Trackingtechnologien von magnetisch[RJC01] bis hin zu inertial[ACD+10] genutzt. In dieser Arbeit wurde das optische Tracking mit der Kamera der Wiimote (siehe Kapitel 2.3.2) und drei selbst entwickelten Sendern für den medizinischen Gebrauch kombiniert.

Als Gegenstände für den täglichen Gebrauch in der Chirurgie wurden ausgewählt: Maske, Brille und Kopfbedeckung (siehe Abbildung 3.7). Jede dieser Konfigurationen wurde mit einer LED Modell IR17-21C von der Firma Everlight, einem Vorwiderstand von 2 Ohm und einer Batterie mit einer passenden Durchlassspannung von 1.5V ausgestattet. Hierbei wurde darauf geachtet, dass die LED einen möglichst hohen Abstrahlwinkel (ca. 120 Grad) besitzt. Damit wurde sichergestellt, dass trotz des eingeschränkten Sichtfeldes (23 Grad vertikal, 33 Grad horizontal) der Kamera die Lichtquelle noch erkannt wurde. Während der Konstruktion der Befestigungen für die LEDs wurde ebenfalls auf den geeigneten Tragekomfort geachtet.

(a) Chirurgische Brille (b) Chirurgische Haube (c) Chirurgische Maske

Abbildung 3.7: Präparierte Gegenstände aus der Chirurgie mit rot markiertem IR Sender

Die Kamera nimmt als Empfänger das IR Signal als Punktquelle in ihrem eigenen Koordinatensystem wahr. Die registrierten Punkte können mit einer x- und y-Koordinate (Einheit: Pixel) von der Wiimote ausgelesen werden. Eine translationale Bewegung besteht aus einem Vektor $\vec{P}$ mit dem Startpunkt P_n und dem Zielpunkt P_m. Die zurückgelegte Distanz von $\vec{P}$ in x/y-Richtung wird für eine horizontale/vertikale Rotation als ein Winkel α/β interpretiert (siehe Formel 3.1).

$$\alpha = x_m - x_n$$

$$\beta = y_m - y_n$$

Formel 3.1: Berechnung der Winkel durch Punktkoordinaten

In der Wirklichkeit resultiert eine Kopfbewegung nach rechts/oben vor einem stillstehenden Objekt in eine Ansicht auf die rechte/obere Seite des Gegenstandes (siehe Abbildung 3.8).

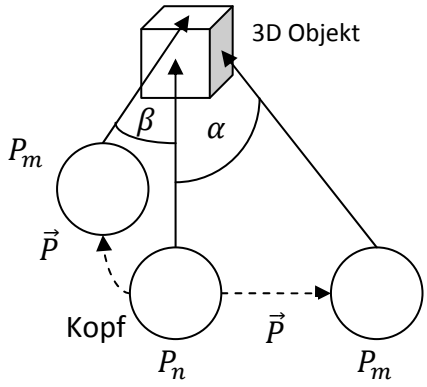


Abbildung 3.8: Perspektivenwechsel in der Realität beim Headtracking

Durch eine Kombination von vertikaler und horizontaler Rotation der virtuellen Kamera kann jede Perspektive auf ein dreidimensionales Objekt dargestellt werden. Jedoch muss die Richtung der Rotation mit dem Winkel β für eine korrekte Abbildung der Realität invertiert werden. Die Begründung dafür ist das fixe Koordinatensystem innerhalb der Wiimote Kamera.

Der Controller wurde konzipiert, um auf den IR Sender zu zeigen und dann eine Bewegung auszuführen. Werden die Positionen vertauscht, verändert sich auch die Orientierung. Bei diesem Vorgang werden die Werte der x-Achse invertiert, aber die Werte der y-Achse bleiben gleich (siehe Abbildung 3.9).

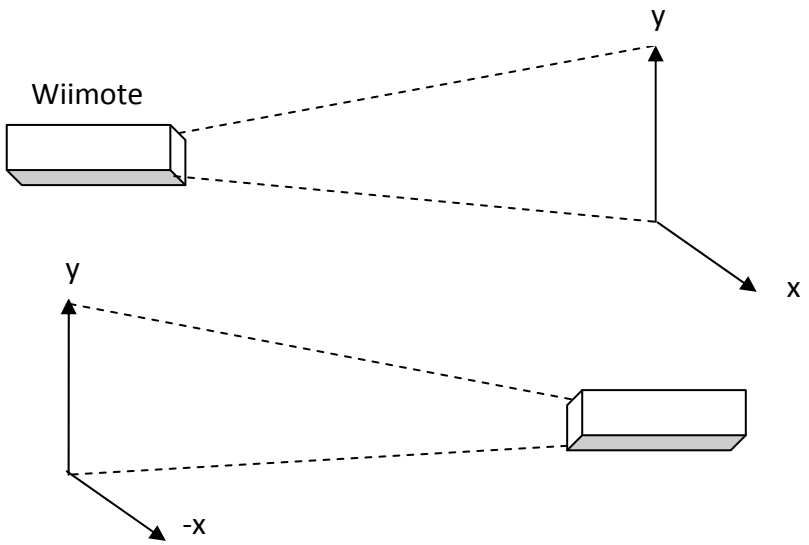


Abbildung 3.9: Ausrichtung des Wiimote Koordinatensystems bei Positionswechsel

Mehrfachrotation für Operationsplanungen

Für eine Operationsplanung ist auch wichtig, dass durch eine schrittweise Bewegung das Objekt beliebig oft gedreht werden kann. Mit dem normalen Headtracking erweist sich das als besonders schwierig, da es zu unangenehmen Kopfbewegungen für den Benutzer kommt. Deshalb wird für eine Operationsplanung der IR Sender durch einen IR Stift ersetzt und ein Grenzwert für aufeinander folgende Bewegungen eingeführt.

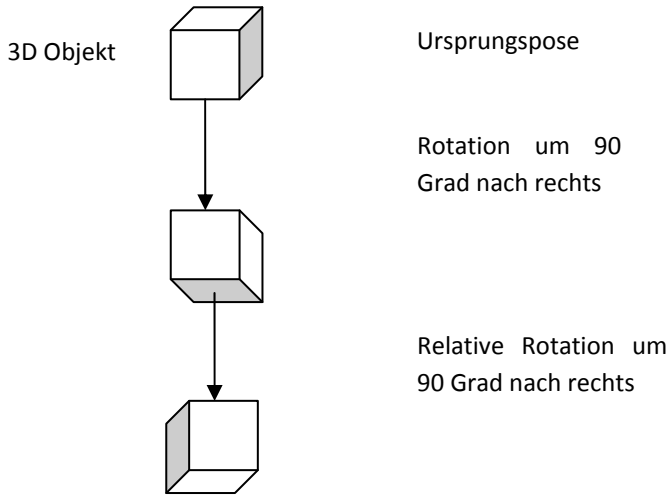


Abbildung 3.10: Mehrfachrotation relative zur aktuellen Pose

Zu Beginn wird der IR Stift aktiviert und eine geradlinige Bewegung durchgeführt. Daraufhin ändert sich die Perspektive auf das Objekt. Wird der IR Sender deaktiviert und überschreitet vor dem nächsten Aktivieren den definierten zeitlichen Grenzwert, so wird das vom System erkannt. Bei erneuter Aktivierung des IR Stiftes gefolgt von einer Bewegung wird eine relative Rotation auf dem Objekt ausgeführt. Das heißt der Bezugspunkt für die Berechnung der Rotation ist stets die aktuelle Pose (siehe Kapitel 2.7) und nicht ursprüngliche Pose des Objektes (siehe Abbildung 3.10).

3.2.2 Kalibrierung

Das VR Headtracking soll in der Lage sein, sich auf unterschiedliche Distanzen zwischen IR Sender und der Kamera einstellen zu können. Deshalb wurde zur Kalibrierung ein Mechanismus eingeführt, der eine Änderung der Sensitivität der Kamerabewegung erlaubt. Zur Realisierung dieses Prozesses wurde das Headtracking in zwei Modi aufgeteilt. Im Modus 1 kann das Headtracking genutzt werden und in Modus 2 wird das Gerät kalibriert. Dies wurde mit zwei Zuständen einer Zustandsmaschine realisiert.

Als *Container* für die einzelnen Zustände wurden die Zustandsautomaten von MITK genutzt[Weg03]. Eine Zustandsmaschine (*state machine*) definiert konkrete Zustände (*states*), die Verhaltensmuster beschreiben. Innerhalb eines Zustand kann es mehrere Übergänge (*transition*) zum selben oder anderen Zuständen geben. Der Übergang von einem Zustand zum nächsten wird durch Ereignis (*event*) gesteuert und kann gleichzeitig eine oder mehrere Aktionen (*action/s*) auslösen. Im weiteren Verlauf wird mit folgendem Schema ein Zustand beschrieben:

state:

- event / transition: action

Abbildung 3.11: Schema für die Beschreibung eines Zustands in einem Zustandsautomaten

Abbildung 3.12 stellt eine grafische Repräsentation der 2 Modi dar. Weiterhin soll es dabei helfen sowohl den Wechsel zwischen den Modi besser zu verstehen als auch welche Funktionen der jeweilge Modus besitzt.

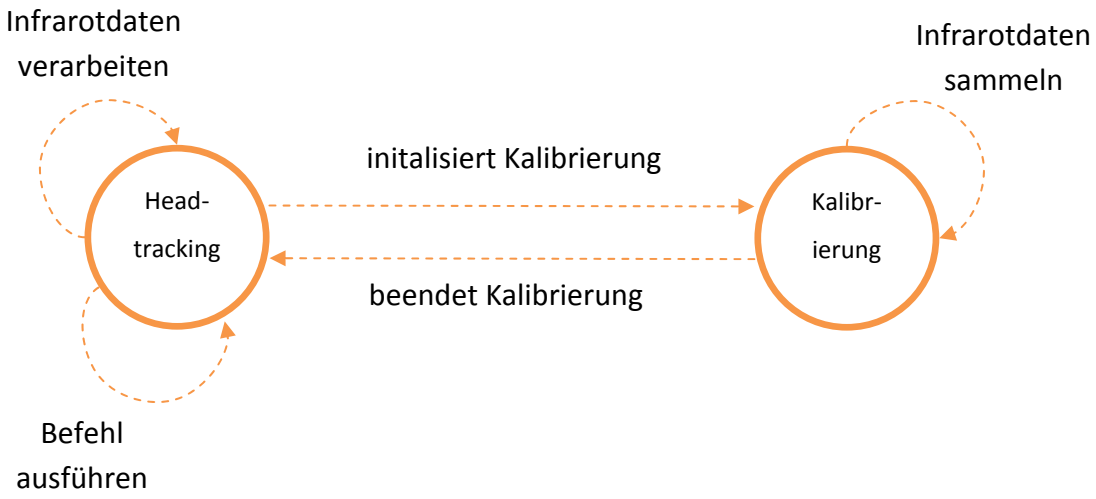


Abbildung 3.12: StateMachine-Pattern für die Kalibrierung des VR Headtracking

Headtracking:

- Empfangen eines Infrarotsignals von der Wiimote / im Zustand bleiben: Bewegungsvektor berechnen und Veränderung der virtuellen Kamera ausführen
- Empfangen eines Buttonsignals von der Wiimote / im Zustand bleiben: Zugewiesenen Befehl ausführen (z.B. Trennen der Verbindung der Wiimote, Reset der Kameraperspektive)
- Empfangen eines Kalibrierungssignals von der Wiimote / Übergang zu Kalibrieriung: Deaktivieren der normalen Funktionalität

Kalibrierung:

- Empfangen eines Infrarotsignals von der Wiimote / im Zustand bleiben: Infrarotdaten sammeln (Minima und Maxima für x- und y-Achse detektieren)
- Empfangen eines Kalibrierungssignals von der Wiimote / Übergang zu Headtracking: Berechnung der neuen Sensitivität mit den gesammelten Daten und daraufhin Reaktivierung der normalen Funktionalität

Der Kalibrierungsprozess setzt sich aus zwei Teilschritten zusammen. Erst werden die vom Benutzer gesendeten IR Daten gesammelt und davon das Minimum x_{min}/y_{min} und Maximum $x_{max}/y_{\max}$ für die x/y-Achse gespeichert. Bei der Eingabe der IR Daten spielt die Form der Bewegungen keine Rolle solange der Benutzer damit sein gewünschtes Sichtfeld beschreibt (siehe Abbildung 3.13).

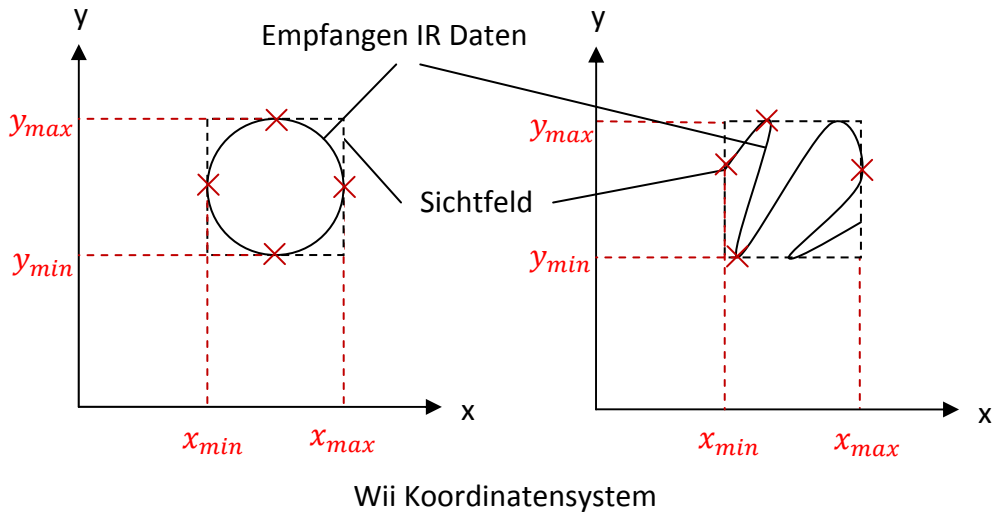


Abbildung 3.13: Bestimmung von Minima und Maxima bei Kalibrierung

Daraufhin wird die Differenz der maximalen horizontalen Auflösung der Kamera $x_{K\,max}/y_{K\,max}$ und der minimalen horizontalen/vertikalen Auflösung der Kamera $x_{K\,min}/y_{K\,min}$ mit der Differenz der gemessenen Kalibrierungswerte x_{max}/y_{max} und x_{min}/y_{min} ins Verhältnis gesetzt (siehe Formel 3.2). Dieser Quotient wird als Skalierungsfaktor mit der aktuellen Sensitivität S_x/S_y für die x/y-Achse multipliziert.

$$S_{Kalibriert\ x} = S_x * \frac{(x_{K\ max} - x_{K\ min})}{(x_{max} - x_{min})}$$

$$S_{Kalibriert\ y} = S_y * \frac{(y_{K\ max} - y_{K\ min})}{(y_{max} - y_{min})}$$

Formel 3.2: Berechnung der Kalibrierungsfaktoren für das Headtracking

Bei einem kleineren Sichtfeld als 1024x768 Pixel im Wii Koordinatensystem erhöht sich demnach die Geschwindigkeit für eine Kamerabewegung.

3.2.3 Transversales Scrollen

Die transversalen Schichtbilder spielen sowohl bei der präoperativen Planung als auch bei der intraoperativen Behandlung eine wichtige Rolle. Damit werden Läsionen (Verletzungen) oder Karzinome (erkranktes Gewebe) in der anatomischen Struktur eines Menschen lokalisiert. Deswegen wurde zur parallelen Verwendung neben dem Headtracking ein transversales Scrollen implementiert. Mit einem IR Stift können 2D Schichtbilder nach oben oder nach unten gescrollt werden (siehe Abbildung 3.14). Ein Aktivieren des IR Sender beginnt eine Eingabe des Benutzers und durch Abschalten wird das Ende der Eingabe signalisiert.

Abbildung 3.14: Scrollen durch transversale Schichtbilder im OP mit rot markierten IR Sender und Strahlenverlauf zur IR Kamera

Wird ein IR Sender innerhalb des Sichtfeldes der Wiimote Kamera geradlinig nach oben bewegt, so wird eine Geste vom System erkannt. Dadurch werden die Schichtbilder vertikal nach oben gescrollt bis die IR Quelle nicht mehr sichtbar ist oder das obere Ende des Bilderstapels erreicht ist. Dabei ist nach es nicht notwendig nach einer erkannte Geste die Linie weiter zu zeichnen, sondern genügt es die aktuelle Position zu halten und für eine aktivierte IR Quelle zu sorgen. Außerdem ist die Erkennung der Geste im Wii Koordinatensystem unabhängig vom Start- und Endpunkt der gezeichneten Linie, weil relative Bewegungen erkannt

werden. Für ein Scrollen vertikal nach unten im Bilderstapel wird eine invertierte Richtung und identischer Betrag der geradlinigen Geste verwendet.

3.2.4 System im OP Einsatz

Das VR Headtracking wurde mit dem Ziel realisiert, im OP-Saal eingesetzt zu werden. Es wurde in Zusammenarbeit mit dem Universitätsklinikum Heidelberg ein reelles Szenario im OP-Saal nachgestellt (siehe Abbildung 3.15). Damit sollte ein Proof of Concept durchgeführt werden und grundsätzliche Funktionalität unter OP-Bedingungen getestet werden.

Aufbau des VR Headtracking im OP-Saal

Zunächst wurde die Wiimote (siehe Abbildung 3.15(1)) an einem Monitor (siehe Abbildung 3.15(2)) für die Anzeige der medizinischen Daten fixiert. Jetzt kann der Chirurg (siehe Abbildung 3.15(3)) mit der IR Brille (siehe Abbildung 3.15(5)) und dem IR Stift (siehe Abbildung 3.15(6)) mit den Schichtbildern als auch der Volumenvisualisierung auf dem OP-Monitor interagieren während er den Patient (siehe Abbildung 3.15(4)) behandelt.

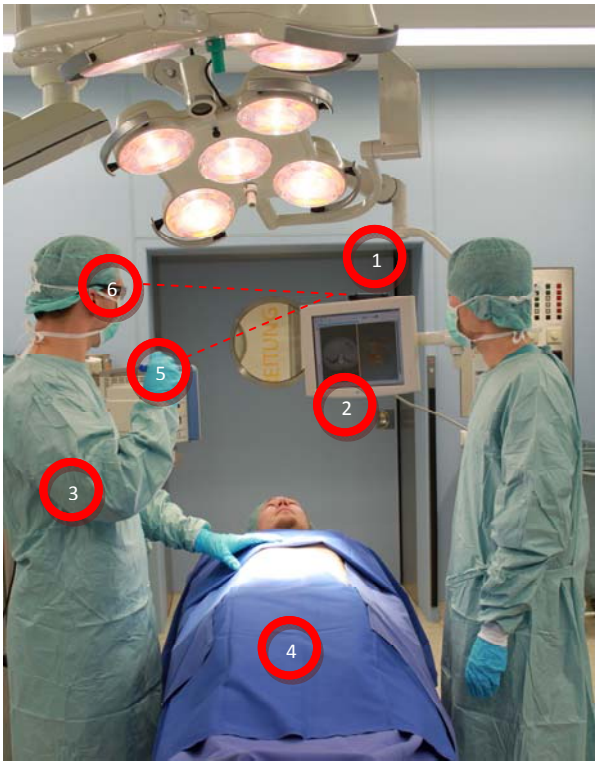


Abbildung 3.15: (1) Fixierte Wiimote (2) OP-Monitor (3) Behandelnder Arzt (4) Patient (5) IR Stift für transversales Scrollen mit Strahlenverlauf (6) Chirurgische Brille mit IR Sender für das Headtracking mit Strahlenverlauf

Es entstand ein dreidimensionaler Eindruck bei Kopfbewegungen des Benutzers und es konnten präzise bestimmte transversale Schichtbilder ausgewählt werden. Auch der Tragekomfort für alle drei IR Sender wurde bestätigt.

Grenzen des Systems

Die Kombination Headtracking und transversales Scrollen konnte erfolgreich in einer Distanz von 1-2m zwischen Benutzer und IR Kamera genutzt werden. Wird dieser Bereich überschritten sind die LEDs nicht mehr stark genug und können nur noch unzuverlässig von der IR Kamera wahrgenommen werden. Das Resultat sind ruckartige Bewegungen in der virtuellen Realität. Während den Untersuchungen führten nur Reflektionen der chirurgische Brille durch die Deckenleuchten im OP-Saal zu Fehleingaben (siehe Abbildung 3.16).

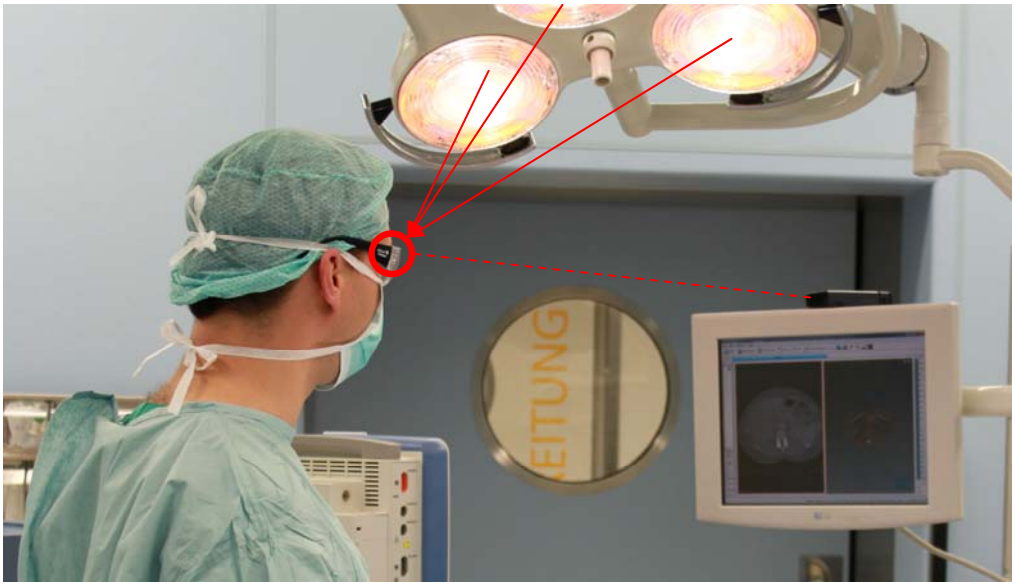


Abbildung 3.16: OP Szenario mit chirurgischer Schutzbrille, rot markiertem IR Sender und Strahlenverlauf. Zusätzliche IR Strahlen von Lichtquellen sind mit roten Pfeilen dargestellt.

Für die erfolgreiche Erfüllung von vordefinierten Aufgaben wurde bereits von Chris R. et. al. [RJC01] eine Evaluation durchgeführt. Es wurde geprüft, wie schnell bestimmte Aufgaben von einem Benutzer mit Headtracking oder mit Maus und Tastatur absolviert werden konnten. Die Ergebnisse zeigen, dass bei der Erledigung der Aufgaben der Gebrauch von Maus und Tastatur gleich schnell oder schneller als das Headtracking ist. Bei dieser Evaluation wurde aber nicht bedacht, dass in der Medizin noch eine weitere Person für die Ausführung solcher Aufgaben notwendig ist. Es wird deshalb vorgeschlagen in einer weiteren Evaluation den Nutzen im medizinischen Umfeld zu testen.

Außerdem berichteten 73% der Tester aus der Evaluation von Chris R. et al. [RJC01] den Eindruck einer komfortableren Bedienung festgestellt zu haben. Das erhöht die Akzeptanz für die Einführung und Verwendung eines solchen Systems im klinischen Umfeld.

Für die VR ist ebenfalls die Genauigkeit der Abbildung der realen Welt von Bedeutung. Im Vergleich mit dem Polhemus Patriot[19], einem 6DoF Motion Tracker, wurden bereits Genauigkeitsmessungen mit der Wiimote von Yang-Wai Chow[Cho09] durchgeführt. Die Resultate zeigten Standardabweichungen von 0.78cm/0.82cm für eine Translation in x/y-Richtung

[19] http://www.polhemus.com/?page=Motion_Patriot, Stand: 09-02-2011

bei einer Detektion IR Sendern durch die IR Kamera der Wiimote. Für den Interaktionsbereich ist das eine akzeptable Abweichung, denn diese geringen Abweichungen werden dem Benutzer bei der Ausführung nicht auffallen.

3.3 Interaktion mit 3D Objekten

Nachdem das virtuelle Headtracking näher betrachtet worden ist, wird an dieser Stelle eine weitere Möglichkeit für die Verbesserung der Interaktion in der Medizin vorgestellt: die direkte Interaktion mit dreidimensionalen Objekten in der VR durch die Wiimote in der Hand des Benutzers und deren integrierte Beschleunigungssensoren als Steuerung.

Ein Ziel dieser Arbeit war es deshalb, die reellen Sensordaten so zu übersetzen, dass eine intuitive Bewegung erzeugt werden kann. Eine Bewegung setzt sich aus zwei Komponenten zusammen, die in Kapitel 3.3.1 beschriebene Rotation und in Kapitel 3.3.2 erläuterte Translation. Die Veränderung der Orientierung (Rotation) wurde mit Quaternionen (siehe Kapitel 2.6) der VNL Bibliothek (siehe Kapitel 2.4.4) umgesetzt. Für die Positionsänderung (Translation) wurde anhand physikalischer Gesetze[TW04] eine Kompensation der Erdbeschleunigung entwickelt und implementiert.

Daraufhin wird eine Bewegung als vollständige Pose in Kapitel 3.3.3 betrachtet. Durch Matrizenmultiplikationen und Posentransformationen (siehe Kapitel 2.7) wurden zwei Modi für eine Interaktion mit 3D Objekten entwickelt. Abschließend werden in Kapitel 3.3.4 die Messungenauigkeiten der Sensoren analysiert und deren Auswirkung auf eine vollständige Darstellung in der VR erläutert.

3.3.1 Rotation

Viele Visualisierungstoolkits verwenden die Orientierung von virtuellen Objekten in einer Rotationsmatrix. Um die Daten der Gyroskope der Wii MotionPlus (siehe Kapitel 2.3.4) in einer virtuellen Szene im MITK nutzen zu können, müssen deshalb mehrere Rechenschritte angewendet werden.

Schritt 1: Integration von Winkelbeschleunigungen

Winkelgeschwindigkeiten werden mit ω und dem entsprechenden Index p für Pitch (Rotation um die x-Achse), r für Roll (Rotation um die y-Achse) und y für Yaw (Rotation um die z-Achse) gekennzeichnet. Der Winkel $\alpha/\beta/\gamma$ steht für den Winkel bei einer Drehung um die x/y/z-Achse. Damit Winkel für eine Rotation berechnet werden können, müssen die Winkelgeschwindigkeiten über die Zeit integriert werden (siehe Formel 3.3).

$$\alpha = \int \omega_p \, dt$$

$$\beta = \int \omega_r dt$$

$$\gamma = \int \omega_y dt$$

Formel 3.3: Einfache Integration von Winkelbeschleunigungen

Schritt 2: Darstellung als Quaternion

Die Winkel α, β, γ können von ihrer Eulerdarstellung in jeweils ein eigenes Quaternion (siehe Kapitel 2.6) überführt werden. Zur Vereinfachung wird im Folgenden nur mit Einheitsquaternionen gerechnet, d.h. die Länge jedes Quaternions entspricht 1. Ein Quaternion wird mit q und passendem Index für die Rotation um die jeweiligen Achsen (x, y, z) versehen. Imaginärteile werden mit i, j und k bezeichnet. θ ist der Winkel für eine Rotation um die durch a_x, a_y, a_z definierte Achse (siehe Formel 3.4).

$$q = \cos\frac{\theta}{2} + \sin\frac{\theta}{2} * (ia_x + ja_y + ka_z)$$

Formel 3.4: Eine allgemeine Definition für ein Quaternion

Durch Einsetzen der in Schritt 1 berechneten Winkel α, β, γ als Radianten θ, φ, τ werden die relevanten Rotationen einzeln als Quaternion dargestellt (siehe Formel 3.5).

$$q_x = \cos\frac{\theta}{2} + \sin\frac{\theta}{2} * (ia_x + 0 + 0)$$

$$q_y = \cos\frac{\varphi}{2} + \sin\frac{\varphi}{2} * (0 + ja_y + 0)$$

$$q_z = \cos\frac{\tau}{2} + \sin\frac{\tau}{2} * (0 + 0 + ka_z)$$

Formel 3.5: Einsetzen der berechneten Parameter in die allgemeine Definition eines Quaternion

Die Kombination von Einzelrotationen in Formel 3.6 erlaubt es mehrere Rotationen, die in beliebiger Reihenfolge ausgeführt wurden, durch ein Quaternion abzubilden.

$$q_{xyz} = q_z * q_y * q_x$$

Formel 3.6: Darstellung von mehreren Rotation mit einem Quaternion

Schritt 3: Einsetzen in eine Rotationsmatrix

Innerhalb von MITK können Rotationsmatrizen genutzt werden um die Orientierung von geometrischen Figuren im Weltkoordinatensystem darzustellen. Veränderungen wie das neu errechnete Quaternion q_{xyz} müssen dementsprechend in eine Matrix umgeformt werden. Zur Vereinfachung wird q_{xyz} substituiert (siehe Formel 3.7).

$$q_{xyz} = q_0 + iq_1 + jq_2 + kq_3$$

Formel 3.7: Substituierte allgemeine Formel für ein Quaternion

Dann werden die Werte in die Formel 2.9 für eine Darstellung eines Einheitsquaternions in einer Rotationsmatrix (siehe Kapitel 2.6) eingesetzt (siehe Formel 3.8).

$$R = \begin{pmatrix} (q_0^2 + q_1^2 - q_2^2 + q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_2q_1 + q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_3q_1 - q_0q_2) & 2(q_3q_2 + q_0q_3) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{pmatrix}$$

Formel 3.8: Rotationsmatrix aus einem Quaternion

3.3.2 Translation

Eine weitere Komponente für eine vollständige Pose ist die Translation. Für die Abbildung der Translationskomponenten die Daten vom Accelerometer der Wiimote (siehe Kapitel 2.3.3) verwendet, da diese für die Detektion von linearen Beschleunigungen geeignet sind. Weil eine Translation in einem dreidimensionalen Raum oft durch einen Vektor repräsentiert wird, werden die Messwerte für translationale Beschleunigung schrittweise in Vektordarstellung überführt.

Schritt 1: g-Kompensation der Messwerte

Die reelle Beschleunigung $\vec{a}$ wird durch die Erdbeschleunigung $\vec{g}$ verändert (siehe Abbildung 3.17). Das gemessene Resultat $\vec{a}_g$ kann also mit den einzelnen Komponenten $\vec{a}$ und $\vec{g}$ charakterisiert werden [TW04].

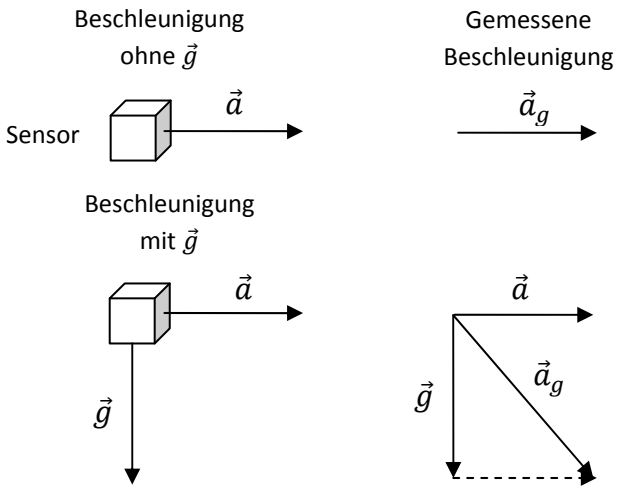


Abbildung 3.17: Beeinflussung der gemessenen Beschleunigung durch Erdbeschleunigung

Um die Beschleunigung $\vec{a}$ berechnen zu können, muss von der gemessenen Beschleunigung $\vec{a}_g$ die Erdbeschleunigung $\vec{g}$ abgezogen werden (siehe Formel 3.9).

$$\vec{a} = \vec{a}_g - \vec{g}$$

Formel 3.9: Allgemeine Formel für die Zusammensetzung der Beschleunigung

Da es sich bei diesem Accelerometer um ein 3 Achsen Sensor handelt, kann für jede Achsenrichtung der wirkende Anteil der gesamten Erdbeschleunigung $\vec{g}$ bestimmt werden. Dazu werden die einzelnen Vektoren in ihre Komponenten für die x-, y-, z-Achse zerlegt (siehe Formel 3.10).

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \begin{pmatrix} a_{gx} - g_x \\ a_{gy} - g_y \\ a_{gz} - g_z \end{pmatrix}$$

Formel 3.10: Darstellung der allgemeinen Formel mit Vektorkomponenten

Die einzelnen Bestandteile g_x, g_y, g_z werden nur von der Orientierung des Sensors beeinflusst, da die Positionsänderung im Raum keine Wirkungsänderung von $\vec{g}$ hervorruft. Die Orientierung setzt sich aus drei Winkeln zusammen. Der Winkel $\theta/\varphi/\tau$ bezeichnet eine Veränderung der Orientierung durch einen Pitch/Roll/Yaw. Weil die Richtung und der Betrag von $\vec{g}$ fest definiert sind, werden zwei Bedingungen festgelegt (siehe Formel 3.11 und Formel 3.12).

$$\theta = \varphi = \tau = 0 = g_x = g_y \rightarrow g_z = |\vec{g}| = 1$$

Formel 3.11: Anfangsbedingung für die Parameter

Da ein Yaw um den Winkel τ keine Veränderung für g_i bedeuten, haben nur die Winkel θ und φ einen Einfluss auf g_x, g_y, g_z. Der Grund dafür ist, dass eine Drehung um die Achse in der $\vec{g}$ wirkt die Lage des Accelerometers verändert, aber der Betrag und die Richtung der existierenden Kräfte auf die Sensoren gleicht bleibt.

$$\forall i \epsilon \{x, y, z\} g_i(\tau) = const.$$

Formel 3.12: Festgelegte Bedingung für Auswirkung eines Yaw auf die Erdbeschleunigung

Weiterhin werden die einzelnen Bestandteile g_x, g_y, g_z im Detail erläutert und definiert. Alle verwendeten Winkel werden als Radianten in die Kosinus- und Sinusfunktionen eingesetzt. Abbildung 3.18 gibt einen Überblick über die Winkelbeziehungen.

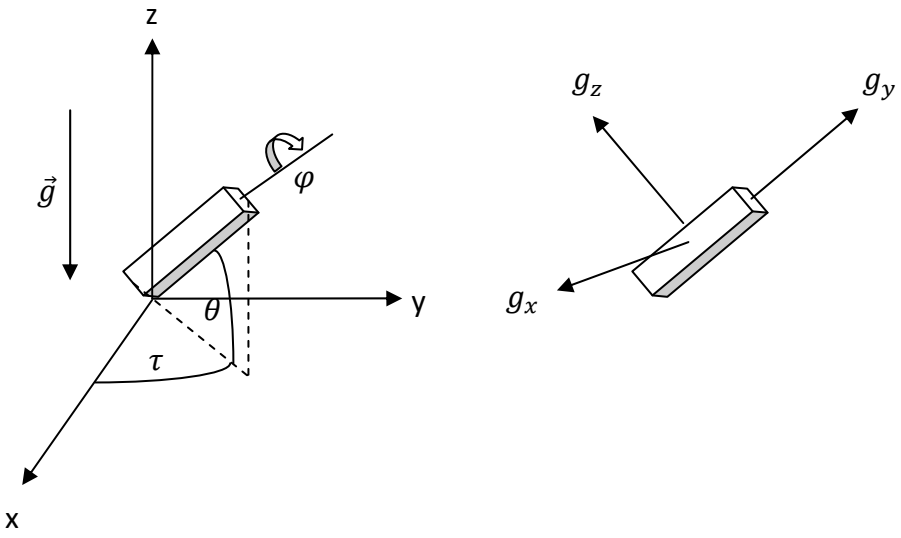


Abbildung 3.18: Winkelbeziehungen zu den Sensoren im Accelerometer

Die erste Komponente g_x wird durch einen Pitch/Roll um den Winkel θ/φ verändert. Umso größer $|\theta|$ wird, desto mehr nähert sich der Sensor für g_x einer Parallelstellung mit $\vec{g}$ an. Aber nur Beschleunigungen die orthogonal zum Sensor auftreten können registriert werden, d.h. mit steigendem $|\theta|$ sinkt der Anteil von $\vec{g}$ für g_x. Umgekehrt wirkt sich ein steigendes $|\varphi|$ aus. Dadurch wird der Sensor in die orthogonale Stellung zu $\vec{g}$ gebracht bis $\vec{g}$ vollständig durch g_x abgebildet werden kann. Durch Parametrisierung von g_x folgt:

$$g_x = \cos\theta * \sin\varphi * |\vec{g}|$$

Formel 3.13: Berechnung der x Komponente von g

Die zweite Komponente g_y wird nur durch einen Pitch um den Winkel θ verändert, weil ein Roll um die zu messende Achse keine Auswirkung auf die Messung selbst hat. Mit steigendem $|\theta|$ nähert sich der Sensor der Lage orthogonal zu $\vec{g}$ an. Deswegen erhöht sich g_y entsprechend bis g_y die Erdbeschleunigung vollständig darstellt. Aufgrund von festgelegten

Ausgaben der Wiimote wird ein Vorzeichenwechsel für den Winkelparameter θ angewendet (siehe Formel 3.14).

$$\sin(-\theta) = -\sin(\theta)$$

$$g_y = -\sin\theta * |\vec{g}|$$

Formel 3.14: Berechnung der y Komponente von g

Die dritte Komponente g_z wird durch einen Pitch/Roll um den Winkel θ/φ verändert. Durch ein steigendes $|\theta|/|\varphi|$ nähert sich der Sensor einer Parallelstellung zu $\vec{g}$ an und damit sinkt der Anteil von $\vec{g}$ der durch g_z gemessen wird. Durch eine Parametrisierung von g_z folgt:

$$g_z = \cos\theta * \cos\varphi * |\vec{g}|$$

Formel 3.15: Berechnung der z Komponente von g

Schritt 2: Zweifache Integration der Sensordaten

Der zurückgelegte Weg x, y, z für eine Translation um die jeweilige Achse wird über eine zweifache Integration über die Zeit erreicht. Die Beschleunigungen werden mit a und den passenden Indizes für die Achsen des Accelerometers bezeichnet (siehe Formel 3.16).

$$x = \iint a_x \; dtdt$$

$$y = \iint a_y \; dtdt$$

$$z = \iint a_z \; dtdt$$

Formel 3.16: Zweifache Integration der gemessenen Linearbeschleunigungen

Schritt 3: Erstellung eines Translationsvektor

Um die Beschleunigung als Translation anwenden zu können, werden die erhaltenen Messwerte mit Hilfe eines Vektors dargestellt. Dazu wird die in Schritt 1 berechnete Zerlegung der Vektoren mit parametrisierten Komponenten g_x, g_y, g_z verwendet (siehe Formel 3.17).

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \begin{pmatrix} a_{gx} - \cos\theta * \sin\varphi * |\vec{g}| \\ a_{gy} - (-\sin\theta * |\vec{g}|) \\ a_{gz} - \cos\theta * \cos\varphi * |\vec{g}| \end{pmatrix}$$

Formel 3.17: Parametriersierte Darstellung der Vektorkomponenten

Daraufhin werden wie in Schritt 2 beschrieben a_x, a_y, a_z zweimal integriert. Das Ergebnis sind die Translationskomponenten x, y, z. Diese können in eine Pose P eingesetzt werden

und ermöglichen mit Hilfe der Orientierung (siehe Kapitel 3.3.1) eine vollständige Repräsentation einer Pose für ein Objekt (siehe Formel 3.18).

$$P = \begin{pmatrix} R_{11} & R_{12} & R_{13} & x \\ R_{21} & R_{22} & R_{23} & y \\ R_{31} & R_{32} & R_{33} & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Formel 3.18: Allgemeine Formel für Pose mit Rotations- und Translationskomponenten

3.3.3 Pose

Für die Interaktion mit 3D Objekten wurden zwei Modi entwickelt. Diese sollen dem Benutzer die Möglichkeit bieten, eine für ihn intuitive Bewegung des Objektes auswählen zu können. Modus 1 führt jede Bewegung relativ zum Objekt aus, während Modus 2 eine Bewegung relativ zur aktuellen Sicht des Benutzers auf das Objekt (Kameraperspektive) ermöglicht (siehe Abbildung 3.19).

Alle Bewegungen werden direkt auf dem 3D Objekt ausgeführt und nicht durch eine Kamerabewegung realisiert, weil bei einer Änderung der Kameraperspektive sich sonst die Ansicht auf alle Objekte in der virtuellen 3D Szene verändern würde. Anhand einer einfachen Rotation soll der Unterschied dieser Modi verdeutlicht werden. Für dieses Beispiel wird davon ausgegangen, dass die Wiimote auf den Bildschirm gerichtet ist.

Im MITK Koordinatensystem befindet sich ein Objekt in seiner Ausgangslage. Nachdem die Bewegung aus dem Wii Koordinatensystem in die Rotationsmatrix R aus Schritt 3 umgeformt worden ist, kann diese auf das aktuelle Objekt einmal für Modus 1 und einmal für Modus 2 angewendet werden (siehe Abbildung 3.19).

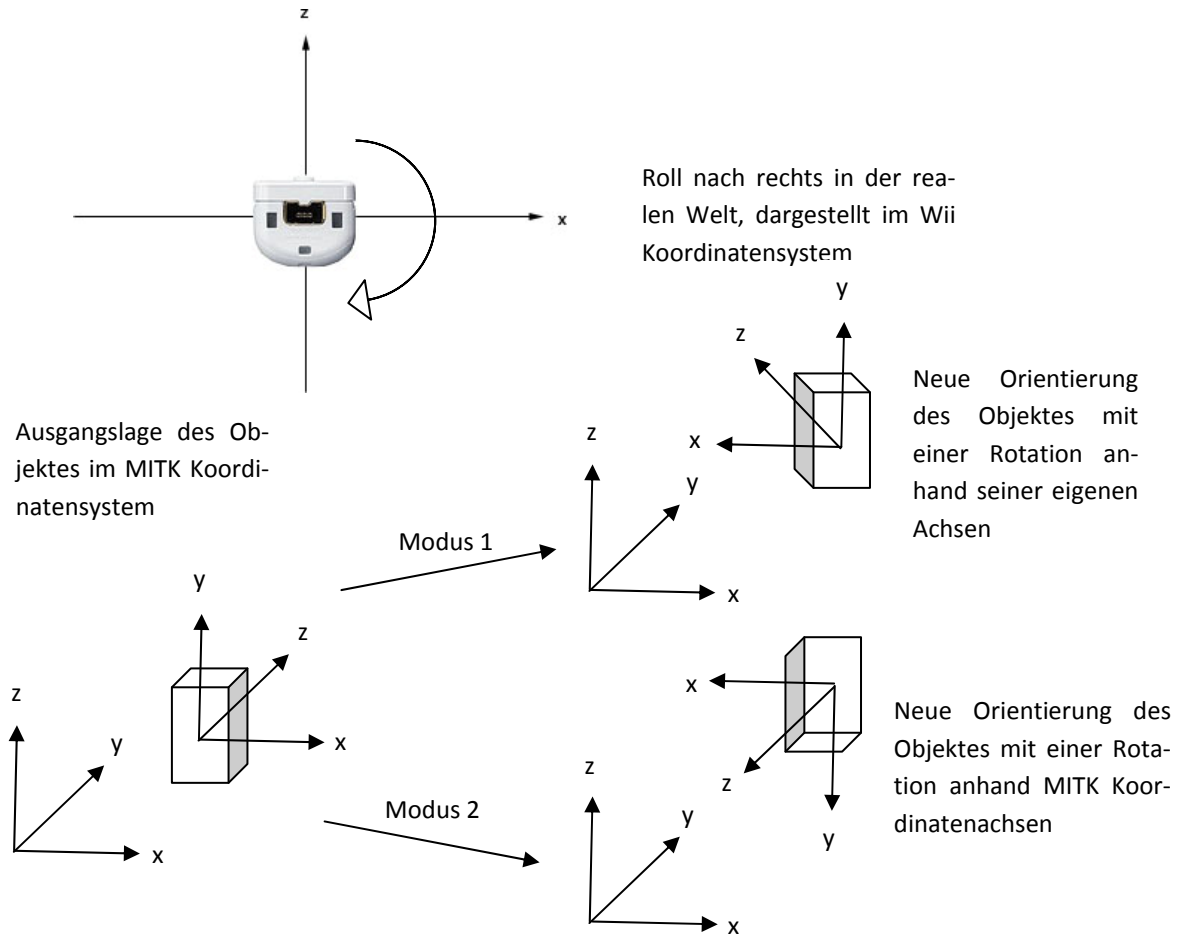


Abbildung 3.19: Rotation mit Modus 1 und Modus 2

Je nach verwendetem Modus wird eine unterschiedliche Koordinatentransformation auf dem aktuellen Objekt durchgeführt. Diese Transformationen können mit Hilfe von Matrixprodukten und Posen eindeutig beschrieben werden (siehe Kapitel 2.7). Eine Pose P eines Objektes setzt sich aus Position und der Orientierung des Objektes zusammen. Sie wird als 4x4 Matrix dargestellt und ist eine Transformation vom Ursprung des Koordinatensystems. In der Matrixdarstellung ist R mit entsprechendem Index die Rotationskomponenten (Orientierung) und T_x, T_y, T_z die Translationskomponenten (Position).

$$P = \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Formel 3.19: Matrixdarstellung einer Pose mit Rotations- und Translationskomponenten

Eine Posentransformation T, ebenfalls wie P definiert, wird durch Matrixmultiplikation auf eine Pose angewandt - das Ergebnis stellt die neue Pose dar. Dabei wird zwischen einer Änderung ΔT von der alten Pose zur neuen Pose eines Objekts und einer Änderung der Pose eines Objektes zu der Pose eines anderen Objektes durch $T_{Von\ Objekt}^{Nach\ Objekt}$ unterschieden.

Die Pfeile in Abbildungen zum Kontext Posentransformationen kennzeichnen bei Transformationen die Richtung und bei Posen den Bezugspunkt. Eine Umkehrung der Pfeilrichtung wird durch die Verwendung der Inversen umgesetzt und zur Vereinfachung nicht zusätzlich mit in Abbildungen aufgenommen. Die Pfeildarstellung dient dazu eine Pose oder Transformation durch andere Posen und/oder Transformationen ersetzen zu können.

Als erstes wird diese Änderung der Pose für Modus 1 betrachtet (siehe Abbildung 3.20). P_{alt} charakterisiert die alte Pose und P_{neu} die neue Pose nach der erfolgter Anwendung von ΔT.

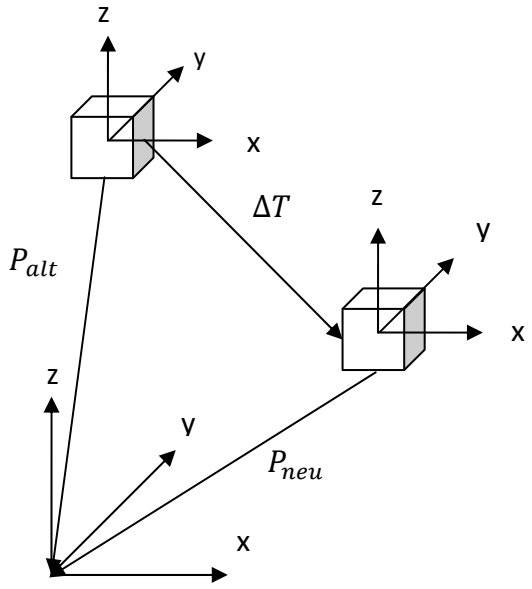


Abbildung 3.20: Transformation relativ zum Objekt

Die Posentransformation ΔT beschreibt im folgenden Verlauf eine rigide Bewegung der Wiimote durch den Benutzer. Die dafür verwendeten Parameter für Orientierung und Position werden aus dem Wii Koordinatensystem für das MITK Koordinatensystem angepasst (siehe Kapitel 3.3.1 und 3.3.2). Zur Bestimmung der Pose P_{neu} können die bekannten Größen P_{alt} und ΔT durch eine einfache Matrixmultiplikation dargestellt werden (siehe Formel 3.20). Die zuerst durchgeführte Transformation wird stets auf die rechte Seite der Multiplikation geschrieben.

$$P_{neu} = P_{alt} * \Delta T^{-1}$$

Formel 3.20: Berechnung der neuen Pose

Als zweites wird die Änderung der Pose für Modus 2 betrachtet. Abbildung 3.21 bietet einen Überblick über die Transformationszusammenhänge und soll dabei helfen die Transformationsformeln besser zu verstehen.

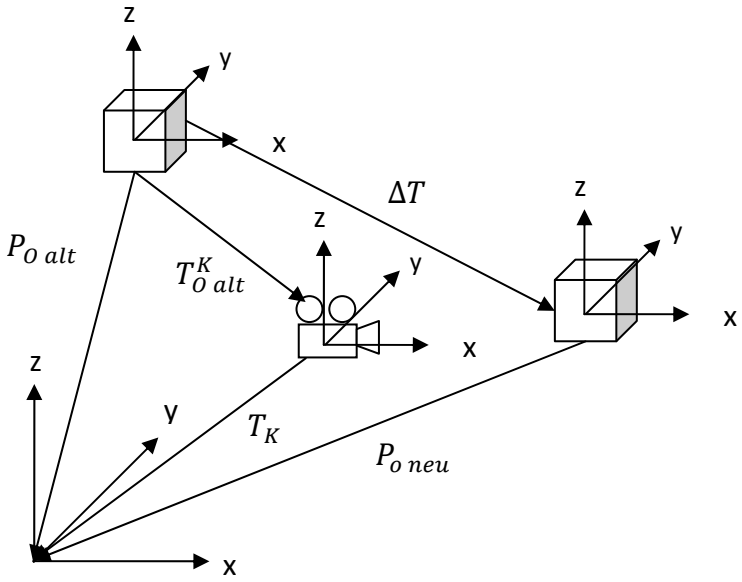


Abbildung 3.21: Transformation relativ zur Kameraperspektive

Damit ΔT aus Kameraperspektive durchgeführt werden kann, wird die Pose $P_{O\ alt}$ auf die Kamerapose P_K mit der Transformation $T^K_{O\ alt}$ abgebildet (siehe Formel 3.21). Dadurch werden die Koordinatenachsen der Kamera auf die Koordinatenachsen des Objektes verschoben.

$$T^K_{O\ alt} = P_K^{-1} * P_{O\ alt}$$

Formel 3.21: Transformation der Koordinatenachse der Kamera auf die Koordinatenachse des Objektes

Daraufhin muss $T^K_{O\ alt}$ auf die aktuelle Pose $P_{O\ alt}$ angewendet werden, um die Transformation durchzuführen. Jetzt wird aus Kameraperspektive die Posentransformation ΔT auf die veränderte Pose des Objektes angewandt. Zum Schluss wird die am Anfang durchgeführte

Transformation $T^{K}_{O\,alt}$ wieder rückgängig gemacht, um die Koordinatenachsen des Objektes wieder in ihren Ursprungszustand zu bringen (siehe Formel 3.22).

$$P^{-1}_{O\,neu} = T^{K^{-1}}_{O\,alt} * \Delta T * T^{K}_{O\,alt} * P^{-1}_{O\,alt}$$

Formel 3.22: Vollständige Transformation der neuen Pose aus Kameraperspektive

Im Gegensatz zum Modus 1 wird hier die Inverse ausgerechnet, da bei diesem Verfahren nicht ΔT auf die aktuelle Pose angewendet wird, sondern nach allen Transformationen $P^{-1}_{O\,neu}$ vom Koordinatenursprung aus durchgeführt.

3.3.4 Messfehler der Wiimote Sensoren

Durch die Bestimmung der Orientierung (siehe Kapitel 3.3.1) und der Position (siehe Kapitel 3.3.2) muss eine vollständige Darstellung der Pose möglich sein, da sich die Pose aus Position und Orientierung zusammensetzt. Jedoch sind die Sensoren des Accelerometers ungenau, sodass Messfehler entstehen. Umso länger diese Messung mit Fehler integriert wird, desto mehr würde sich diese Messungenauigkeit akkumulieren. Die folgende Abbildung 3.22 zeigt die gemessenen Werte bei Ruhelage des Accelerometers.

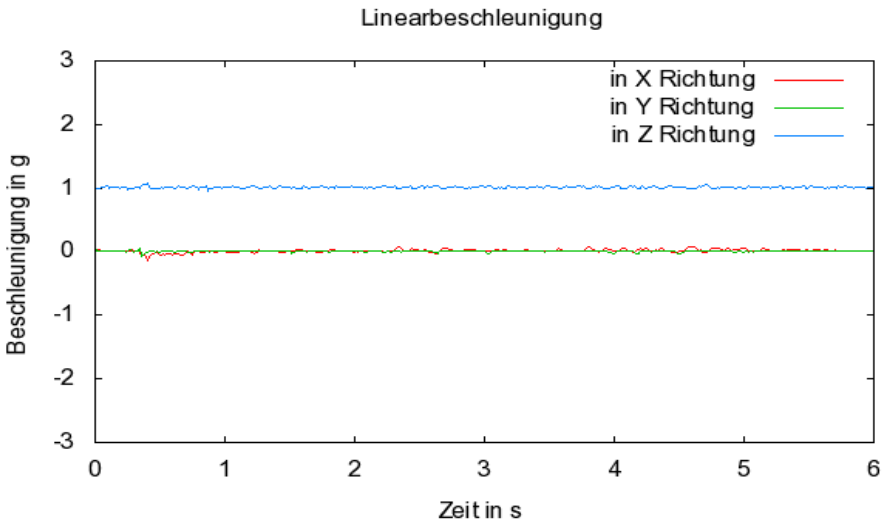


Abbildung 3.22: Messfehler des Accelerometers in Ruhelage

Auch für die Gyroskope in der WiiMotion Plus wurde in einer Evaluation von Yang-Wai Chow[Cho09] zur Genauigkeit festgestellt, dass die verwendeten Sensoren im Vergleich zum Polhemus Patriot[20] eine Standardabweichung von 0.58/1.02/0.70 Grad für yaw/pitch/roll besitzt.

Außerdem kann eine Handbewegung eines Menschen ebenfalls zu Ungenauigkeiten bei der Messung der beiden Sensoren führen. Da dadurch ein Gyroskop/Accelerometer sowohl bei

[20] http://www.polhemus.com/?page=Motion_Patriot, Stand: 09-02-2011

einer Rotation als auch einer Translation beeinflusst wird, kann nicht mehr unterschieden werden, ob die detektieren Information explizit zu einer der beiden Bewegungen gehören. In Kombination mit den Messfehlern, die alleine durch die Sensoren entstehen, erweist es sich als schwierig, die Translation von der Rotation zu trennen.

Mit einer Korrektur der Messfehler der Sensoren durch einen geeigneter Filter (z.B. Kalman Filter, Condensation Algorithmus) wäre eine genauere Abbildung in einer VR und damit eine konkrete Trennung der Rotation und Translation möglich.

Die Anwendung solcher VR Anwendungen spielt eine zentrale Rolle bei der Unterstützung von medizinischen Behandlungen[LMW+08] [SNS+04] und präoperativen Planungen[SMR+09]. Die Ergebnisse in diesem Bereich deuten darauf hin, dass sich diese Technologie zu einem Schlüsselelement für die Steigerung der medizinischen Behandlungsqualität (verkürzte Behandlungsdauer, geringere Fehlerrate, einfachere Bedienung) entwickeln kann.

4 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein flexibles Integrationsdesign für externe Eingabegeräte erarbeitet und implementiert. Durch Verwendung von standardisierten Interfaces und das C++ Framework BlueBerry (siehe Kapitel 2.4.2) war die Umsetzung einer losen Kopplung und Modularität möglich. Außerdem war es durch den Einsatz von Extension Points möglich ein dynamisches Verhalten während der Laufzeit zu realisieren.

Anhand dieses Integrationskonzepts wurden Module in MITK für die Benutzung des Space Navigators (3D Maus, Analog Devices) und die Wiimote (Gamecontroller, Nintendo) entwickelt. Mit dem Space Navigator können so komplexe virtuelle Kamerafahrten mit 6 Freiheitsgraden (siehe Kapitel 2.3.1) durchgeführt werden. Für den Integrationsprozess der Wiimote wurde das Integrationsdesign noch durch Multithreading von Qt (siehe Kapitel 2.4.3) und ITK (siehe Kapitel 2.4.4) erweitert.

Ein VR Headtracking konnte mit Hilfe der IR Kamera (siehe Kapitel 2.3.2) der Wiimote entwickelt werden. Mit drei eigens entwickelten IR Sendern an chirurgischer Maske, Haube und Brille wurden Kopfbewegungen von der Realität in eine VR abgebildet und in konkreten Klassen implementiert. Dort verändern diese Bewegungen die Ansicht auf eine Volumenvisualisierung, die aus medizinischen Daten generiert wurde. Zusätzlich ermöglicht der Einsatz eines IR Stiftes das gleichzeitige Scrollen durch transversale Schichten, um die Aufgaben im prä- sowie intraoperativen Einsatz zu vereinfachen. Das ganze System wurde im OP hinsichtlich seiner Funktionalität mit Erfolg getestet.

Weiterhin wurde die Interaktion mit 3D Objekten durch die Daten der Gyroskope (siehe Kapitel 2.3.4) der WiiMotion Plus realisiert. Dabei wurden zwei unterschiedliche Modi entwickelt, um dem Benutzer eine Auswahlmöglichkeit für eine intuitive Handhabung von Objekten zu verschaffen. Die Änderung einer Pose, die aus Orientierung (Rotation) und Position (Translation) besteht, konnte teils erfolgreich in der VR abgebildet werden. Vor allem eine klare Trennung von Rotation und Translation bei der Detektion durch die Sensoren erwies sich als schwierig. Einerseits ist das bedingt durch Messungenauigkeiten, die bei einer normalen Handbewegung entstehen. Andererseits durch einen Systemfehler des Accelerometers (siehe Kapitel 2.3.3), der ohne die Verwendung eines geeigneten Filters den akkumulierten Messfehler nicht kompensiert.

Es wird vorgeschlagen durch Auswahl und Einsatz eines Kalman Filters oder Condensation Algorithmus die Messungenauigkeit in einer weiteren Studie zu bearbeiten.

5 Diskussion

Motivation und Ziel

Die zunehmende Integration von Computern in die Medizin soll den Behandlungsprozess vereinfachen und die Qualität steigern. Davon sind vor allem drei Bereiche betroffen:

- Die präoperative Operationsplanung,
- Die intraoperative Unterstützung durch medizinische Datensätze
- Trainingssimulationen zur Erlernung neuer OP-Verfahren oder Verbesserung von Routineabläufen standardisierte OP-Verfahren

Die Verarbeitung der medizinischen Information wird oftmals durch Softwarewerkzeuge wie dem MITK realisiert. Dabei erschwert eine hohe Komplexität neuer Computerverfahren die leichte Interaktion mit den Daten. Für die Operationsplanung ist es notwendig, dass das Personal mit dem Umgang der Software geschult wird. Während eines aufwendigen Behandlungsprozesses im OP Saal wird neben dem Arzt eine weitere Person benötigt, die auf Anweisung die richtige Ansicht auf die Daten auswählt. Bei Trainingssimulationen bedeutet eine inkorrekte Abbildung der Realität eine Verringerung des Lerneffekts.

Ziel dieser Arbeit war es, intuitive Interaktionen anhand von low-cost Eingabegeräten (3D Maus - Space Navigator, Wii Contoller) umzusetzen und ins MITK zu integrieren. Dadurch soll es möglich sein, diese Eingabegeräte für die Interaktion mit medizinischen Daten zu verwenden. Der Arzt soll freihändig arbeiten können und gleichzeitig mit den Volumenvisualisierungen interagieren können. Dabei soll eine einfache und zuverlässige Handhabung gewährleistet werden, indem die Daten von den Eingabegeräten korrekt in eine virtuelle Umgebung projiziert werden. Außerdem soll mit der Wiimote als Steuerung eine direkte Interaktion mit 3D Objekten zur Unterstützung des klinischen Behandlungsprozesses möglich sein.

Diskussion der Ergebnisse

Im Rahmen dieser Arbeit wurden die low-cost Eingabegeräte Space Navigator und Wiimote in das MITK eingebunden. Sie wurden verwendet um eine benutzerfreundliche und intuitive Interaktion mit medizinischen Daten zu ermöglichen. Das VR Headtracking stellt eine freihändige Bedienung von Volumenvisualisierungen und bietet parallel das Scrollen durch transversale Schichtbilder. Ebenso die Wiimote als Steuerung selbst erlaubt eine Interaktion mit 3D Objekten.

Zu Beginn wurde dafür ein Integrationsdesign für externe Eingabegeräte erarbeitet und implementiert. Es bietet Modularität und lose Kopplung durch den Einsatz von standardisierten Interfaces. Dadurch wird eine leichte Wartung und Wiederverwendbarkeit ermöglicht. Weiterhin konnte durch die Verwendung von Extension Points und XML Dateien ein dynamisches Verhalten während der Laufzeit realisiert werden. Weil es auf dem C++ Framework

BlueBerry basiert, würde eine Wiederverwendung automatisch in einer zusätzlichen Abhängigkeit resultieren. Jedoch wurde BlueBerry auf dem anerkannten OSGi Standard und dem Konzept der Extension Points von Eclipse umgesetzt. Deshalb können diese Prinzipien leicht auf ein anderes System übertragen werden.

Bei der Integration von Space Navigator sowie Wiimote wurden Treiber verwendet, die nur unter dem Betriebsystem Windows (XP, Vista, 7) funktionieren. Deswegen ist die Nutzung der Geräte eingeschränkt. Weil Windows XP für medizinische Geräte sowie in Kliniken immer noch ein weit verbreitetes Betriebssystem ist, stellt es für die Verwendung momentan kein Hindernis dar. Zudem ist es bei der Wiimote sogar notwendig, dass Pfadangaben zum WDK und SDK statisch in die Entwicklungsumgebung eingebunden werden müssen. Jedoch gibt es für die Wiimote wie den Space Navigator noch Treiber, die auf anderen Betriebssystemen funktionieren. So kann ein Portierbarkeit auf andere System gewährleisten werden.

Das durch die Kamera der Wiimote erstellte VR Headtracking ermöglicht eine freihändige Interaktion mit 3D Oberflächen für die Navigation und das Scrollen durch transversale Schichtbilder zur Lokalisierung von Läsionen (Verletzungen am Gewebe) oder Karzinomen (erkranktes Gewebe). Diese 3D Objekte sind aus medizinischen Daten konstruiert und können in verschiedenen Bereichen genutzt werden.

Das aktuelle VR Headtracking bedeutet aber auch eine Reduktion der Eingabemöglichkeiten für den Benutzer, da neben dem VR Headtracking nur ein transversales Scrollen möglich ist. Für das mit dem Universitätsklinikum ausgearbeitete OP Szenario ist das ausreichend und wird zusätzlich durch die IR Kamera der Wiimote beschränkt. Die Verwendung von neueren Eingabegeräten mit innovativen Technologien könnte eine vollständige Steuerung der VR ermöglichen. Trotz der verlängerten Ausführung bei den Aufgaben, könnte durch die Intuivität einer simplen Kopfbewegung die Akzeptanz und Schulungszeiten für die Handhabung der Software gesenkt werden. Das Ergebnis einer Evaluation[RJC01] ist, dass 73% der Tester den Eindruck einer komfortableren Bedienung im Gegensatz zu Maus und Tastatur hatten. Das erhöht die Akzeptanz für die Verwendung durch das klinische Personal.

Von technischer Seite liegt die Abweichung bei Messgenauigkeit der IR Kamera im Vergleich zu einem teuren Trackingssystem (Polhemus Patriot)[Cho09] von 0.78 und 0.82 cm abhängig von der gemessen Achse. Weil für die Anwendung eines VR Headtracking keine hohe Präzision notwendig ist, wird die Messabweichung in diesem Zusammenhang als vernachlässigbar betrachtet.

Außerdem hat sich bei einem Proof of Concept im OP herausgestellt, dass ein solches VR Headtracking im OP einsetzbar ist. Jedoch ist die Distanz für eine korrekte Funktionalität auf 1-2m beschränkt und unter Umständen durch die Verwendung eines ungeeigneten IR Senders (Chirurgische Brille führt zu Reflektionen von IR Licht) zusätzlich gestört werden kann. Die chirurgische Brille wird selten im klinischen Umfeld genutzt, deshalb spielt das Reflektionsproblem eine untergeordnete Rolle.

Die IR Sender wurden so entwickelt, dass sich nach erfolgtem Einsatz an eine neue Brille/Maske/Haube angebracht werden können und die Energiequelle einfach gewechselt werden kann. Ebenfalls durch den Einsatz einer besser geeigneten LED oder sogar eines komplett Setup für das optische Tracking könnte die Distanz erhöht und Reflektionen verhindert werden.

Neben dem VR Headtracking ist die Interaktion mit 3D Objekten durch Verwendung von Gyroskopen in der WiiMotion Plus und des Accelerometers in der Wiimote entwickelt worden. Zur besseren Steuerung gibt es zwei Modi, die dem Benutzer die Wahl einer präferierten Änderung des 3D Objektes im Raum erlauben. Modus 1 führt eine Bewegung relativ zur der Lage des 3D Objektes aus, während Modus 2 als Referenz für eine Bewegung die Kameraperspektive nutzt.

Unabhängig von der Darstellung im virtuellen Raum muss der Benutzer in der realen Welt die Wiimote nach vorne gerichtet auf den Bildschirm bedienen. Da die Wiimote durch ihre äußere Form bereits eine solche Haltung impliziert, ist die Verwendung für den Benutzer dennoch intuitiv. Des Weiteren kann das Konzept so weiterentwickelt werden, dass das System unabhängig von der Haltung in der realen Welt funktioniert. Dazu können ebenfalls Quaternionen in Kombination mit den Beschleunigungssensoren verwendet werden.

Je nach Modus kann eine Pose des Controllers aus der Realität in der VR abgebildet werden. Eine Pose setzt sich zusammen aus Orientierung (Rotation) und Position (Translation). Zur Abbildung der Rotation werden die gemessenen Daten der Gyroskope schrittweise transformiert und können somit in die Software übertragen werden.

Für die Translation werden die Informationen des Accelerometers verwendet. Wegen Messungenauigkeiten, die bei der Benutzung der Wiimote in der Hand eines Menschen entstehen, ist eine Trennung von Translation und Rotation nur schwer möglich. Es muss ein geeignetes Modell entworfen werden, dass die Handbewegungen und daraus resultierenden Messwerte für die Sensoren genau beschreibt. Außerdem sorgen ungenaue Sensoren im Accelerometer dafür, dass eine Translation nicht korrekt in die VR abgebildet werden kann. Dieses Problem kann durch den Einsatz eines geeigneten Filters wie eine Variante des Kalman Filters oder des Condensation Algorithmus in einer weiteren Studie bearbeitet werden.

Auch bei Genauigkeitsmessungen[Cho09] wird ersichtlich, dass eine Standardabweichung von 0.58/1.02/0.70 Grad für den yaw/pitch/roll für die Simulation eines intraoperativen Szenarios ungeeignet ist. Jedoch könnten sich Trainingssimulationen als eine Entwicklungsmöglichkeit für Ärzte und zur Evaluation solcher Verfahren für den klinischen Einsatz als nützlich erweisen. Auch wenn der Lerneffekt geringer ausfällt, als bei einer sehr genauen Simulation, tritt trotzdem ein Lerneffekt ein. Zudem wird durch eine OP-Simulation die Patientensicherheit nicht gefährdet im Gegensatz zu Absolventen, die direkt an einem realen Patienten trainieren müssen.

Diese Mixed Reality Anwendungen[LMW+08][SNS+04] eignen sich für klinische Anwendungen. Jedoch wird oftmals nicht die benötigte Präzision erreicht, damit eine benutzerfreundliche und intuitive Interaktion für eine OP Simulation gewährleistet ist. Da sich diese Technologien in der Medizin noch ihren Anfängen befinden, könnte durch eine präzisere und teurere Technik und der dazugehörigen Forschung der intraoperative Einsatz in der Zukunft realisieren lassen.

Zusammenfassend lässt sich sagen, dass bereits mit low-cost Eingabegeräten wie dem Space Navigator und der Wiimote die Schlüsselrolle von intuitiven Interaktionskonzepten in der Medizin für die Zukunft zeigen lässt. Durch die Kombination Tracking und Interaktion kann mit einfachen Mitteln eine freihändige Steuerung für medizinische Daten entwickelt werden. Dadurch wird die Akzeptanz des klinischen Personals erhöht, die Gefährdung für den Patienten gesenkt und gleichzeitig die Lernzeit standardisierter Verfahren verkürzt.

6 Ausblick

Das in dieser Arbeit entworfene Konzept kann weiterhin in MITK genutzt um weitere externe Eingabegeräte zu integrieren. Ein weiteres Projekt kann die Kinect von Microsoft sein, die durch ihre neue Technik eine Gestenerkennung erlaubt. Dadurch können vorhandene Beschränkungen der Wiimote wie die Reduktion der Eingabemöglichkeiten aufgehoben werden und durch geeignete Interaktionskonzepte erweitert werden.

Neben Eingabegeräten aus der Spieleindustrie kann auch Hardware aus der Medizintechnik eingesetzt werden. Dadurch können bestehende Entwicklungen wie das VR Headtracking oder die Interaktion von 3D Objekten angepasst werden und gleichzeitig deren Präzision erhöht werden. Außerdem kann der Vergleich von bereits vorhandenen Entwicklungen im Bereich Interaktion mit neuen Projekten wie der Kinect gezogen werden.

Der bereits vorhandene Treiber der Wiimote kann gegen einen Treiber ausgetauscht werden, der mit alle anderen Betriebssystem kompatibel ist. Dann können bereits entwickelte Interaktionen mit der Wiimote auch für die breite Masse von Benutzern zugänglich gemacht werden. Weiterhin können für Treiber standardisierte Interface eingeführt werden, sodass eine Wartung und Erweiterung von bereits vorhandenen Grundprinzipien der Interaktion einfacher wird.

Das bereits erstellte VR Headtracking und die Interaktion mit 3D Objekten können einerseits in Kombination mit Traininungssimulation für Ärzte modifiziert werden. Damit können beide Interaktionen zu Schulungszwecken von alten und neuen Verfahren getestet werden. Andererseits können weitere Konzepte wie ein Stereokamerasystem mit zwei Wiimotes, Finger Tracking und Gestensteuerung für Interaktionen entwickelt werden.

Durch ein Stereokamerasystem kann ein schnelles Prototyping für zeitnahe Messergebnisse als weitere Bewertungsgrundlage durchgeführt werden. Das Finger Tracking ist ein Konzept, das bereits in den alltäglichen Gebrauch durch Touchpads umgesetzt wird. Damit können auch in der Medizin Interaktionen mit medizinischen Daten intuitiver gestaltet werden. Die Gestensteuerung würde die Reduktion von Eingabemöglichkeiten aufheben und ein Vielzahl von freihändigen Eingabemöglichkeiten ermöglichen.

Abkürzungen

6DoF	6 Degrees of Freedom
AR	Augmented Reality
BlueBerry	Modular, cross-platform, C++ application framework
Bundle	MITK Package mit BlueBerry dependency
CMake	Cross-platform build system
FoV	Field of View
GUI	Graphical User Interface
HCI	Human-Computer-Interaction
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
IR	Infrarot
ITK	Insight Segmentation and Registration Toolkit
MCI	Mensch-Computer-Interaktion
MITK	Medical Imaging Interaction Toolkit
Module	MITK Package
MR	Mixed Reality
OS	Operating System
SDK	Software Development Kit
SOA	Service Oriented Architecture
QT	Cross-plattform application and UI C++ framework
VR	Virtual Reality
VTK	Visualization Toolkit
WDK	Windows Driver Kit
Wiimote	Wii Controller

Anhang

A Tutorial zur Integration eines Eingabegerätes

Dieser Abschnitt beschreibt in Detail den Integrationprozess des Space Navigator und stellt ein Tutorial dar. Durch das neue Design besteht ein Eingabegerät mindestens aus den folgenden semantischen Teilen: Treiber, Event, Add-on und Controller. Das Add-on stellt einen Eventhandler dar und erlaubt eine Weiterleitung neu erzeugter Events vom Treiber zum Controller. Dort können interpretiert und transformiert werden in tatsächliche Aktionen. Im Moment ist es nicht möglich, diesen Prozess zu entwickeln ohne den Source Tree von MITK zu ändern.

- Jedes Add-on muss das Interface EventMapperAddOn implementieren, damit es selbst zum Eventmapper hinzugefügt werden kann und eine Weiterleitung zum MITK Core möglich ist
- Passenden Konstanten für alle möglichen Events und Aktionen müssen erstellt werden

Wichtige Notizen

1. Beide Konstantentypen müssen in der Datei mitkInteractionConst.h definiert sein
2. Weil die Controller Klasse eine Statemachine ist, muss der Parameter, der dem Konstruktor übergeben wird (z.B. MyInputDeviceInteraction) in beide StateMachine.xml Dateien eingefügt werden und diesen befinden sich in den folgenden Verzeichnissen:

   MITK-SRC-DIR/mitk/Core/Code/ Interactions
   MITK-SRC-DIR/mitk/CoreUI/Bundles / org.mitk.gui.qt.common/ resources

   Ansonsten kann die Statemachine durch Events vom Eingabegerät keine Zustandsprünge machen.

- Add-on müssen die Eventkonstanten benutzen um Events vom Treiber zum Eventmapper weiterreichen zu können
- Die Controller Klasse wandelt die Informationen, die von selbst erzeugten Aktionskonstanten kommt, in Aktionen um: beispielsweise eine Änderung der Perspektive der virtuellen Kamera

Modul

Als erstes muss ein Verzeichnis, das die implementierten Klassen enthält, mit einem passenden Namen (z.B. MyInputDevice) im Modulverzeichnis erzeugt werden. Es könnte auch nur ein externe Bundle genutzt werden, anstatt der Kombination aus Modul und Bundle. Aber bei Beachtung des Prinzips der losen Kopplung würde ein Bundle allein eine zusätzliche Abhängigkeit zu BlueBerry bedeuten. Es müssen die folgenden Dateien verändert werden:

- files.cmake:

```
SET(CPP FILES
    MyInputDevice/mitkMyInputDeviceAddOn.cpp
    MyInputDevice/mitkMyInputDeviceDriver.cpp
    MyInputDevice/mitkMyInputDeviceEvent.cpp
    MyInputDevice/mitkMyInputDeviceController.cpp
)
```

- CMakeLists.txt:

  in dieser Datei muss der Name des Moduls, die include Verzeichnisse (d.h. Unterverzeichnisse) und die Abhängigkeiten definiert werden

```
MITK CREATEMODULE(mitkMyInputDevice
INCLUDE DIRS MyInputDevice
DEPENDS Mitk
)
```

Wichtige Notizen

Dieses Modul wird nicht gebaut, es sei denn ein anderes aktiviertes Bundle oder Modul hat eine Abhängigkeit auf das Modul.

Extension Point

Das Bundle org.mitk.core.ext stellt mit der ID org.mitk.core.ext.inputdevices einen Extension Point bereit, der von anderen Bundles dazu verwendet werden die Liste von bekannten Eingabegeräten zu erweitern. Zum Hinzufügen eines neuen Eingabegerätes muss die Datei plugin.xml vom Bundle nach folgendem Schema editiert werden:

```
<?xml version ="1.0" encoding="UTF-8"?>
    <plugin>
        <extension point="org.mitk.core.ext.inputdevices">
            <inputdevice id="org.mitk.inputdevices.myinputdevice"
            name="MyInputDevice" class="mitk::MyInputDeviceActivator">
            <description>short description </description>
            </inputdevice>
        </extension>
</plugin>
```

Das Attribut point vom Tag extension muss zu dem Extension Point mit der ID org.mitk.core.ext.inputdevices verweisen. Ansonsten kann der Extension Point das Eingaberät nicht identifizieren. Das extension Tag folgt festgelegten Regeln, die der Datei inputdevice.exsd des org.mitk.core.ext Bundle zu finden sind. Dieses Schema enthält Beschränkungen für die Struktur eines Tags für Eingabegeräte. So kann ein Hinzufügen eines neuen Eingabegerätes dazu führen, dass die Regeln geändert werden. Bis jetzt ist dieses Schema nur informativ und wird noch nicht validiert. Für ein einfaches Eingabegerät reichen 3 Attribute aus:

- id: einzigartiger Kennung für jedes Eingabegerät
- name: reeller Name des Eingabegerätes
- class: die Klasse zur Aktivierung mit seinem namespace

Die Beschreibung ist obligatorisch und bietet normalerweise eine kurze Zusammenfassung. Der Seite für die Einstellungen wird automatisch ein neues Eingabegerät mit einer Checkbox hinzugefügt und in einer XML Datei gespeichert, wenn bestätigt wird. Zum Beispiel, falls die ExtApp genutzt wird wäre der Pfad:

MeinLaufwerk:/.ExtApp/.BlueBerryPrefs/prefs.xml

Bundle

In CMake muss als Source-Verzeichnis der folgende Pfad gesetzt werden, um den Bundlegenerator zu nutzen: MITK-SRC-DIR/mitk/Build/Tools/BundleGenerator und das Binary-Verzeichnis ist jetzt das Ziel für den eigenen Code. Ein temporäres Verzeichnis sollte außerhalb des Projektes erzeugt werden und als Binary-Verzeichnis ausgewählt werden. Nachdem das Projekt einmal konfiguriert worden ist, müssen die folgenden zwei Variablen gesetzt werden:

- PLUGIN ID: einzigartige Kennung (z.B. org.mitk.inputdevices.myinputdevice)
- PLUGIN NAME: normaler Name, der dem Benutzer anzeigt werden kann

Dann muss das Projekt zweimal konfiguriert werden und daraufhin die Projekt Dateien erzeugt werden. Damit die Extension funktioniert, muss die Datei plugin.xml manuell in das gewünschte Verzeichnis hinzugefügt werden. Der nächste Schritt ist das Erzeugen und Hinzufügen einer Klasse, die alle notwendigen Abhängigkeiten (Hinzufügen der Add-ons zum Eventmapper, Hinzufügen eines Listeners zu dem globalen Eventhandler) regelt. Diese Klasse muss das Interface IInputDevice von dem Bundle org.mitk.core.ext implementieren, um erreichbar zu sein ohne Verwendung einer zusätzlichen Abhängigkeit. Wie bereits in Kapitel Modul erwähnt, muss CMakeLists.txt eine Abhängigkeit auf das Modul haben. Das kann umgesetzt werden durch das Hinzufügen der folgenden Parameter zu dem Macro vom Bundle des Eingabegerätes:

```
MACRO CREATE MITK PLUGIN(mitkMyInputDevice)
```

Am Ende muss das Verzeichnis im temporären Ordner in das Bundle Verzeichnis von MITK oder zu einem eigenen Projekt Ordner verschoben werden.

Treiberkonflikte

Das Problem tritt nur auf, falls ein Modul, wie *mitkInputDevices*, aus mehr als einem Eingabegerät besteht und so mehr als einen Treiber benötigt. Für den Fall, dass nur einer dieser Eingabegeräte eingeschaltet ist, kann ein fehlender Treiber des anderen Gerätes Probleme mit dem Build-Prozess verursachen.

Erklärung der drei Parameter des OPTON Befehls:

```
OPTION(MITK USE MYINPUTDEVICE DRIVER "Use my input devicedriver" OFF)
IF (MITK USE MYINPUTDEVICE DRIVER)
ADD DEFINITIONS(-DMITK USE MYINPUTDEVICE DRIVER)
ENDIF(MITK USE MYINPUTDEVICE DRIVER)
MITK CREATEMODULE(mitkInputDevices
INCLUDE DIRS MyInputDevice
DEPENDS Mitk )
```

- MITK USE MYINPUTDEVICE DRIVER:
  der Name der Option in CMake
- Use my input device driver:
  eine kurze Beschreibung
- OFF:
  der Standardwert für diese Option

Die if-Definition nach dem OPTION Befehl überprüft, ob die Flag aktiviert ist und falls ja, fügt sie allen Code hinzu, der damit entfernt wurde.

Das folgende Beispiel zeigt wie die Option zur Klasse hinzugefügt werden kann:

```
#ifdef MITK USE SPACENAVIGATOR DRIVER
code
#endif
```

Kritischer Code wird nur ausgeführt, wenn der Benutzer sicherstellt, dass der Treiber installiert ist und die Flag eingeschaltet ist.

B USE Cases

Wiimote VR Headtracking

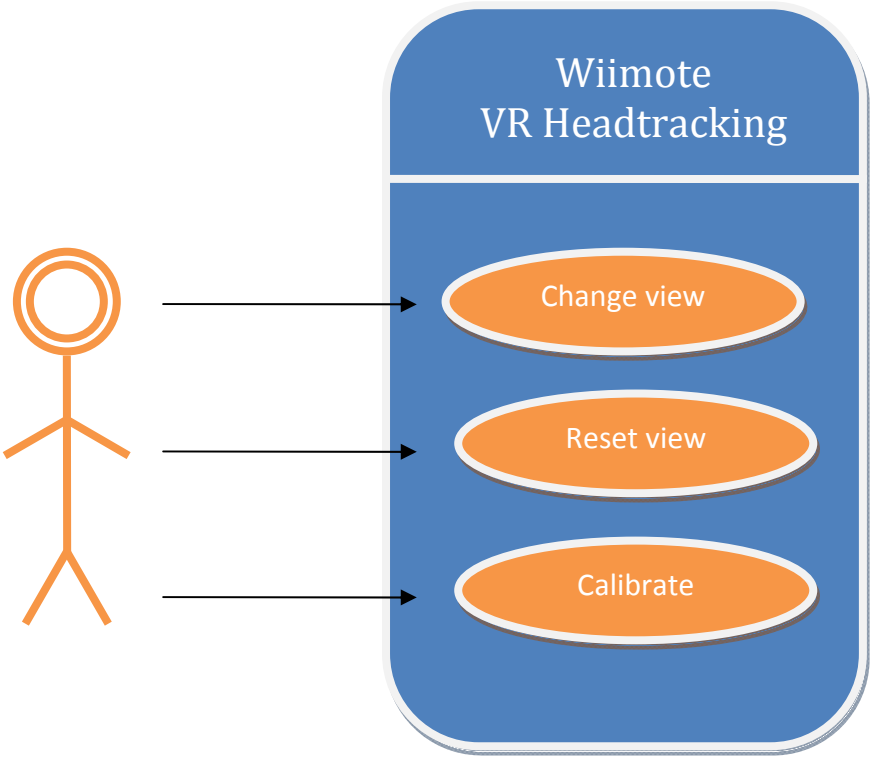


Abbildung B.1: Use case diagram VR Headtracking

Change view

Durch Kopfbewegungen kann die Kameraperspektive auf die medizinischen Daten im MITK verändert werden.

Use Case Details

Name:	Change view
Voraussetzungen:	Wiimote und Headtrackinggerät betriebsbereit, VR Headtracking in MITK aktiviert
Akteure:	Benutzer

Use Case Description

	Akteureingabe	*Systemantwort*
1.	Senden eines IR Signals	
2.		Erkennen des IR Senders durch Wiimote IR Kamera
3.		Umwandeln der IR Daten in Koordinaten
4.		Verwenden der Koordinaten zur Berechnung eines Bewegungsvektors
5.		Ändern der Kameraperspektive entsprechend der Daten
6.		Aktualisieren der Ansicht für den Benutzer

Extension Points

2.1 Falls die IR Quelle nicht erkannt wird, erfolgt keine Änderung der Kameraperspektive und es wird mit 1 fortgefahren.

4.1 Falls die Länge des Bewegungsvektors zu gering ist, wird keine Änderung der Kameraperspektive durchgeführt und mit 1 fortgefahren.

Reset view

Durch Drücken des Home-Buttons auf der Wiimote kann die Kameraperspektive zurückgesetzt werden.

Use Case Details

Name:	Reset view
Voraussetzungen:	Wiimote und Headtrackinggerät betriebsbereit, VR Headtracking in MITK aktiviert
Akteure:	Benutzer

Use Case Description

	Akteureingabe	*Systemantwort*
1.	Drücken des Home-Buttons	
2.		Deaktivieren des VR Headtracking
3.		Zurücksetzen der Kameraperspektive auf die Standardansicht
4.		Reaktivieren des VR Headtracking

Extension Points

Keine vorhanden

Calibrate

Durch Drücken des A-Buttons der Wiimote wird die Kalibrierung gestartet. Daraufhin werden alle IR Signal für die Kalibrierung verwendet. Durch erneutes Drücken des A-Buttons wird der Prozess beendet und die neue Sensitivität für die Kamerabewegung berechnet.

Use Case Details

Name:	Change view
Voraussetzungen:	Wiimote und Headtrackinggerät betriebsbereit, VR Headtracking in MITK aktiviert
Akteure:	Benutzer

Use Case Description

	Akteureingabe	*Systemantwort*
1.	Drücken des A-Buttons	
2.		Deaktivieren des VR Headtracking
3.	Senden der IR Signale	
4.		Sammeln der IR Daten
5.	Drücken des A-Buttons	
6.		Berechnen der neuen Sensitivität mit den gesammelten Daten
7.		Reaktivieren des VR Headtracking

Extension Points

6.1 Falls keine IR Signale empfangen werden konnten, wird dem Benutzer das mitgeteilt und es wird mit 1 fortgefahren

Wiimote Interaktion mit 3D Objekten

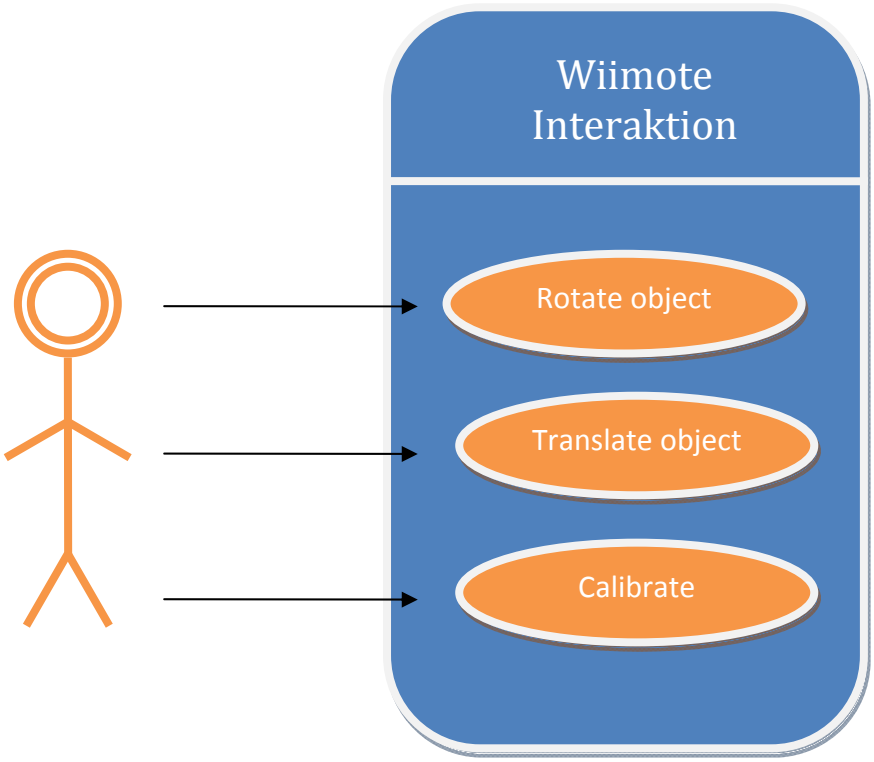


Abbildung B.2: Use case diagramm Interaktion mit 3D Objekten

Rotate object

Während der B-Button der Wiimote gedrückt wird, kann durch eine Rotation der Wiimote eine Rotation eines ausgewählten 3D Objektes im MITK durchgeführt werden.

Use Case Details

Name:	Rotate object
Voraussetzungen:	Wiimote betriebsbereit, Surface Interaction in MITK aktiviert
Akteure:	Benutzer

Use Case Description

	Akteureingabe	*Systemantwort*
1.	Drücken des B-Buttons und halten der Taste	
2.		Aktivieren der Surface Interaction
3.	Rotieren der Wiimote	
4.		Erkennen einer Rotationsbewegung
5.		Transformieren der Daten der Wiimotesensoren zur Verwendung im
6.		Ausführen der Rotationsbewegung mit dem angewählten Objekt
7.	Loslassen des B-Buttons	
8.		Deaktivieren der Surface Interaction

Extension Points

3.1 Falls keine Rotation erkannt werden konnte, wird erst mit 7 und dann mit 1 fortfahren.

4.1 Falls keine Rotationsbewegung erkannt wird, fortfahren mit 3.

5.1 Falls die Daten nicht transformiert werden können, wird der Benutzer durch eine Fehlermeldung darauf aufmerksam gemacht. Es wird mit 7 dann mit 1 fortgefahren.

Translate object

Während der B-Button der Wiimote gedrückt wird, kann durch eine Translation der Wiimote eine Translation eines ausgewählten 3D Objektes im MITK durchgeführt werden.

Use Case Details

Name:	Translate object
Voraussetzungen:	Wiimote betriebsbereit, Surface Interaction in MITK aktiviert
Akteure:	Benutzer

Use Case Description

	Akteureingabe	*Systemantwort*
1.	Drücken des B-Buttons und halten der Taste	
2.		Aktivieren der Surface Interaction
3.	Translatieren der Wiimote	
4.		Erkennen einer Translationsbewegung
5.		Transformieren der Daten der Wiimotesensoren zur Verwendung im
6.		Ausführen der Translationsbewegung mit dem angewählten Objekt
7.	Loslassen des B-Buttons	
8.		Deaktivieren der Surface Interaction

Extension Points

3.1 Falls keine Translation erkannt werden konnte, wird erst mit 7 und dann mit 1 fortfahren.

4.1 Falls keine Translationsbewegung erkannt wird, fortfahren mit 3.

5.1 Falls die Daten nicht transformiert werden können, wird der Benutzer durch eine Fehlermeldung darauf aufmerksam gemacht. Es wird mit 7 dann mit 1 fortgefahren.

Calibrate

Durch Drücken des Plus-Buttons wird der Kalibrierungsprozess gestartet und es werden Messwerte von den Sensoren aufgezeichnet.

Use Case Details

Name:	Calibrate
Voraussetzungen:	Wiimote betriebsbereit, Surface Interaction in MITK aktiviert
Akteure:	Benutzer

Use Case Description

	Akteureingabe	*Systemantwort*
1.	Drücken des Plus-Buttons	
2.		Deaktivieren der Surface Interaction
3.	Senden von Sensordaten in Ruhelage	
4.		Aufzeichnen der Sensordaten in Ruhelage
5.	Drücken des Minus-Buttons	
6.		Berechnen der Abweichungen und Messfehler durch gesammelte Daten
7.		Reaktivieren der Surface Interaction

Extension Points

Keine Vorhanden

C Klassendiagramme

Wiimote Modul

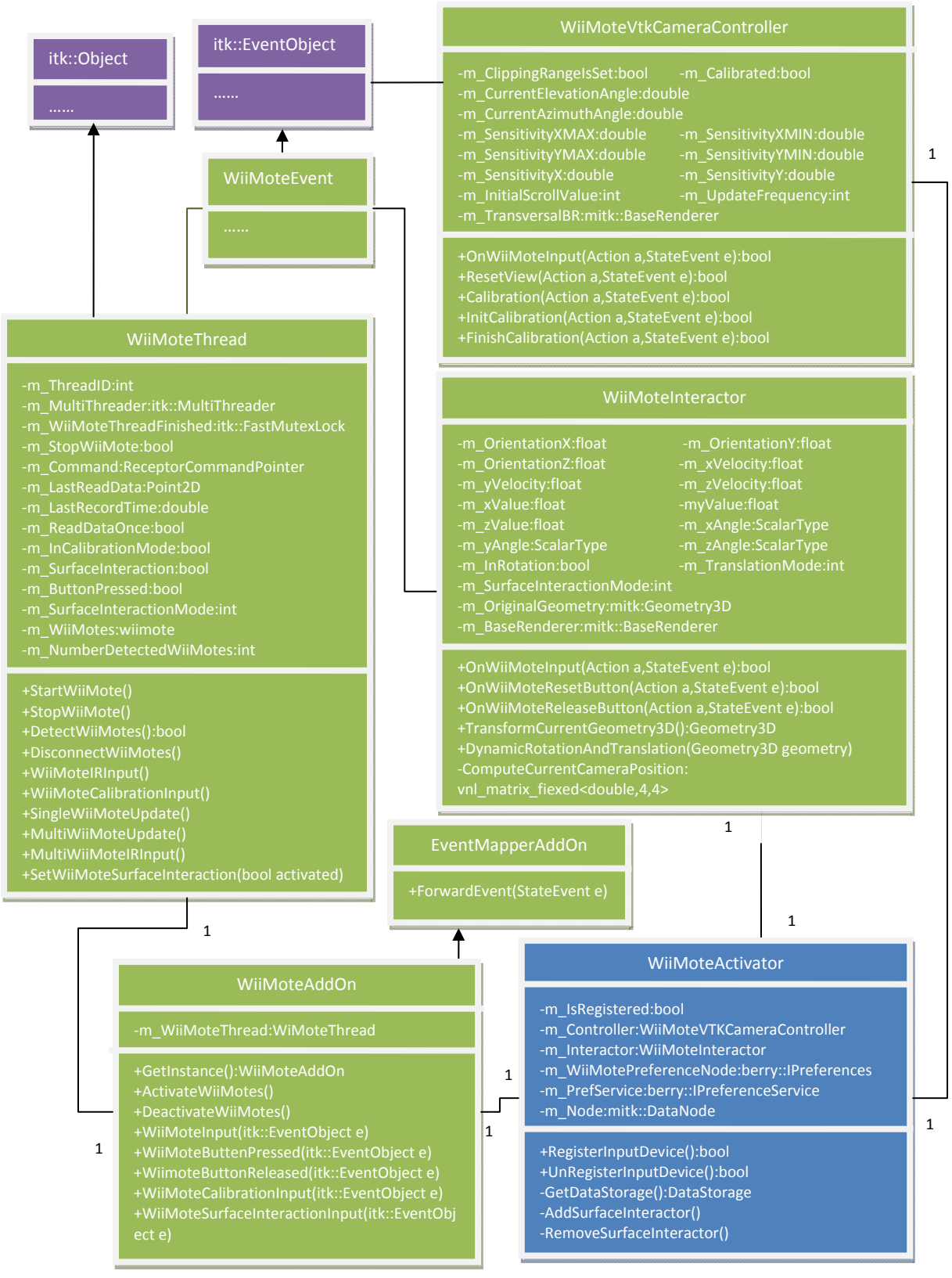


Abbildung B.3: Klassendiagramm Wiimote Modul

Space Navigator Modul

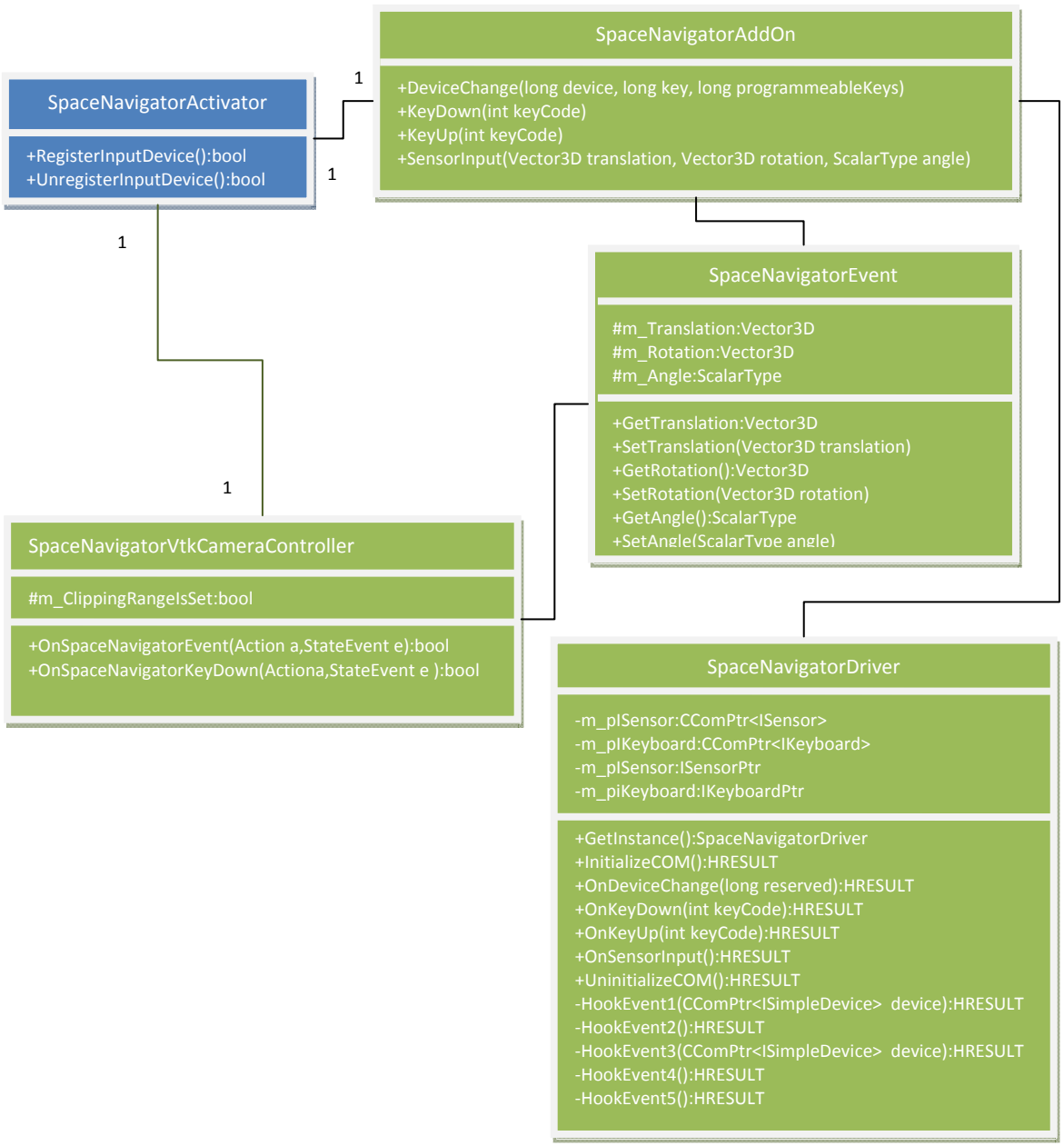


Abbildung B.4: Klassendiagramm Space Navigator Modul

Abbildungsverzeichnis

Formelverzeichnis

Literaturverzeichnis

[Abh09] Björn Abheiden. Entwicklung eines räumlichen Interaktionskonzeptes. Studienarbeit, Hochschule Esslingen, 2009.

[ACD+10] Marion N. Armenise, Caterina Ciminelli, Francesco Dell'Olio, Vittorio M. N. Passaro. Introduction. In: Advances in Gyroscope Technologies. Springer-Verlag, Berlin, Heidelberg, 2010.

[Ana07] Analog Devices. Spezifikation Infrarot Kamera: Model ADXL330. 2007. Quelle: http://www.analog.com/static/imported-files/data_sheets/ADXL330.pdf. Stand: 22-01-2011.

[Bar01] Markus Bartz. Quaternionen. Seminar Computergraphik, Universität Koblenz-Landau,2001.

[Cho09] Yang-Wai Chow. Low-Cost Multiple Degress-of-Freedom Optical Tracking for 3D Interaction in Head-Mounted Display Virtual Reality. In: International Journal of Recent Trends in Engineering, Issue. 1, Vol. 1, 2009.

[Duc05] Tan Tran Druc. Modeling and Simulation of the Capacitive Accelerometer. Diplomarbeit, college of technology, Vietnam National University, Hanoi (VNUH), 2005.

[Eps11] Epson Toyocom. Data sheet Model XV-3500CB Single-Axis Gyroscope. Quelle: http://www.epsontoyocom.co.jp/english/product/Sensor/set01/xv3500cb/index.html. Stand: 22-01-2011.

[FLW09] Jung-Leng Foo, Thom Lobe, Eliot Winer. A Virtual Reality Environment for Patient Data Visualization and Endoscopic Surgical Planning. Technical Report, Journal of Laparoendoscopic & Advanced Surgical Techniques, Volume 19, Supplement 1, 2009.

[Hei04] Andreas M. Heinecke. Erste interaktive Systeme. In: Mensch-Computer-Interaktion. Carl Hanser Verlag, 2004, S.15-16.

[Her05] Michael Herczeg. Werkzeuge. In: Softwareergonomie. Oldenburg Wissenschaftsverlage GmbH, 2.Auflage, 2005, S.2-3.

[HNH08] Simon Hay, Joseph Newman, Robert Harle.Optical Tracking Using Commodity Hardware. In: Proceedings of the 7th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2008.

[Inv08] InvenSense. Data sheet Model IDG-600 Dual-Axis Gyroscope. 2008. Quelle: http://www.datasheet.in/download.php?id=685825. Stand: 22-01-2011.

[ISN+05] Luis Ibánez, Will Schroeder, Lydia Ng, Josh Cates, and the Insight Software Consortium. The ITK Software Guide. Second Edition, 2005.

[Jaz06] Reza. N. Jazar. Global Roll-Pitch-Yaw Angles. In: Theory of Applied Robotics – Kinematics, Dynamics, and Control. Springer Verlag, 2.Auflage, 2006, S.62-63.

[Kor86] Berthold K. P. Korn. Closed-form solution of absolute orientation using unit quaternions. In:Journal of the Optical Society of America A, Vol.4, 1987, 629 ff.

[LD05] Nicola Di Lorenzo und Jenny Dankelman. Surgical Training and Simulation. Report. Global Surgery – Future Directions 2005.

[Len04] Eric Lengyel. Quaternions. In: Mathematics for 3D Game Programming & Computer Graphics. Charles River Media, Inc., 2.Auflage, 2004, S.86-88.

[LJG+08] Micah Lapping – Carr, Odest Chadwicke Jenkins, Daniel H. Grollman, Jonas N. Schwertfeger, Theodora R. Hinkle. Wiimote Interfaces for Lifelong Robot Learning. Association for the Advancement of Artificial Intelligence (AAAI), 2008.

[LMW+08] Christian A. Linte, John Moore, Andrew D. Wiles, Chris Wedlake and Terry M. Peters. Virtual Reality-Enhanced Ultrasound Guidance: A Novel Technique for Intracardiac Interventions. In: Computer aided surgery: official journal of the International Society for Computer Aided Surgery, 2008, 13(2):82-94.

[MMS+10] Daniel Maleike, Michael Müller, Alexander Seitel, Marco Nolden. BVM-Tutorial 2010: Entwicklung interaktiver medizinischer Bildverarbeitungssysteme mit MITK. BVM, 2010.

[Mül10] Michael Müller. The BlueBerry Framework – A closer look on the MITK application framework. MITK Schulung, 20-09-2010.

[Neu10] Jochen Neuhaus. User Interface Programming with Qt4 for MITK Developers. MITK Schulung, 21-09-2010.

[Neß08] Robert Neßelrath. TaKG – Ein Toolkit zur automatischen Klassifikation von Gesten. Masterarbeit, Universität des Saarlandes, 2008.

[RJC01] Chris Raymaekers, Joan de Boeck, Karin Coninx. Assessing Head-Tracking in a Desktop Haptic Environment. In: Usability Evaluation and Interface Design – Cognitive Engineering, Intelligent Agents and Virtual Reality. Lawrence Erlbaum Associate, Inc., 1.Auflage, 2001, S.302-305.

[Rme05] Reducing Physical Discomfort and Pain Among 3D Computer Users. In: VSI Risk Management and Ergonomics, 2005.

[Sch06] Bernd Schwald. Punktbasiertes 3D-Tracking starrer und dynamischer Modelle mit einem Stereokamerasystem für Mixed Reality. Dissertation, Technische Universität Darmstadt, 2006.

[SLH+10] Schaaf T., Lamontain H., Hilbert J., Schilling F., Tolxdorff T. Simulation and training of ultrasound supported anesthesia: a low-cost approach. Medical Imaging 2010: Ultrasound Imaging, Tomography, and Therapy und Proc. of SPIE Vol. 7629.

[SMR+09] M. Seitel, L. Maier-Hein, U. Rietdorf, S. Nikoloff, A. Seitel, A. Franz, H. Kenngott, M. Karck, R. de Simone, I. Wolf, and H.-P. Meinzer. Towards Mixed Reality Environment for Preoperative Planning of Cardiac Surgery. In: Studies in health technologies and informatics, 2009, 142:307-9.

[SNS+04] L. Soler, S. Nicolau, J. Schmid, C. Koehl, J. Marescaux, X. Pennec, N. Ayache. Virtual Reality and Augmented Reality in Digestive Surgery. In: ISMAR '04 Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, 2004.

[TW04] David Titterton und John Weston. Basic principles of strapdown inertial navigation systems. In: Strapdown Inertial Navigation Technology. MPG Books Ltd, 2.Auflage, 2004.

[Weg03] Wegner Ingmar. Entwurf und Realisation eines generischen Interaktionsmodells mit Undo-Funktionalität für die medizinische Bildverarbeitung. Diplomarbeit, Universität Heidelberg / Hochschule Heilbronn, 2003.

[WJ01] Werner Korb und Pierre Jannin. Bewertung der Mensch-Maschine-Interaktion. In: Computerassistierte Chirurgie. Urban & Fischer Verlag, 1.Auflage, 2011, S.323-331.

[Wol07] Jörg Wollnack. Homogene Transformation von Posen. In: Vorlesungsunterlagen zum Thema: Robotik – Analyse, Modellierung und Identifikation. Technische Universität Hamburg-Harburg, 2007, S.1-4.

[WVW+05] Ivo Wolf, Marcus Vetter, Ingmar Wegner, Thomas Böttger, Marco Nolden, Max Schöbinger, Mark Hastenteufel, Tobias Kunert, Hans-Peter Meinzer. The Medical Imaging Interaction Toolkit. In: Medical Image Analysis 9, 2005, 594-604.

[QT10] Nokia. Cross-plattform application and UI C++ framework. Quelle: http://qt.nokia.com/products/, Stand: 16-02-2011.