Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht

Henrik Brauer & Konrad Glugla

Augmented Reality on the iPhone

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Henrik Brauer & Konrad Glugla

Augmented Reality on the iPhone

Projektbericht eingereicht im Rahmen des Master-Projekts
im Studiengang Informatik (Master of Science)
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Kai von Luck

Abgegeben am 28. Februar 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die vorliegende Ausarbeitung behandelt die Themen Augmented Reality in Verbindung mit dem Apple iPhone als Anzeige- und Interaktionsgerät. Dieses Projekt ist Teil der Ausstellung „Emotional Tent“, welches ein Projekt des Masterstudiengangs der Informatik (HAW Hamburg, Wintersemester 08/09) in Kooperation mit dem Department Design ist.

1.1 Motivation und Projektziel

Unter Augmented Reality versteht man die computergestützte Erweiterung der Realitätswahrnehmung. Diese Information kann alle menschlichen Sinnesmodalitäten ansprechen, häufig wird jedoch unter erweiterter Realität nur die visuelle Darstellung von Informationen verstanden.

Augmented Reality könnte in praktisch allen Bereichen des Alltags zum Einsatz kommen. Monteure könnten sich den nächsten Arbeitsschritt direkt in ihr Sichtfeld einblenden lassen, Soldaten oder Katastrophenhelfer könnten sich Ziele und Gefahrenzonen im Gelände anzeigen lassen und Designer könnten mit tatsächlich und virtuell anwesenden Kollegen am selben dreidimensionalen Modell arbeiten. Mit fortschreitender Technologie lassen sich futuristische Anwendungsszenarien erschließen: Elektronische Geräte, die nur virtuell existieren, aber auf echte Berührungen reagieren, künstliche Sinneserweiterungen wie den „Röntgenblick“ und Computerspiele in freiem Gelände.

Die Idee dieses Projekts war es Augemented Reality zu nutzen um einen interaktiven Umgang mit der Umwelt zu ermöglichen. Diese sollte speziell in Hinblick auf die Projektidee der interaktiven Kunst entwickelt werden. Augmented Reality bietet hierbei eine Fülle von Möglichkeiten. Der Grundgedanke war es die Augmented Reality Anwendung in das Konzept des Emotional Tent einzubinden und eine erweiterte Sinneswahrnehmung zu bieten. Augmented Reality bietet vor allem die Chance sehr leicht verschiedene Stimmungen zu erzeugen und Objekte darzustellen die in der realen Welt nicht erstellt werden können.

Die Darstellung der Augmented Reality sollte an mehreren räumlich voneinander getrennten Orten durchgeführt werden. Der Organismus des „Emotional Tent“ sollte dabei nicht durch große Anzeigegeräte die mitten im Raum stehen gestört werden. Viel mehr sollte der Anwender ein mobiles Endgerät bei sich tragen und an entsprechder Stelle in die Augmented Reality „eintauchen“.

Trotz der Mobilitätsanforderung sollte das Display über eine ausreichende Größe verfügen, auf denen die Anzeige von Videos noch eine Akzeptanz beim Anwender findet.

Nach dem heutigen Stand der Technik eignet sich für die zuvor definierten Anforderungen vor allem das iPhone aus dem Hause Apple. Das iPhone wird heutzutage häufig zur Anzeige von Multimediainhalten eingesetzt und erreicht dabei mit seiner Displaygröße von 3.5 inch und einer Auflösung von 480x320 Pixeln eine hohe Akzeptanz. Zudem bietet es intuitive Bedienungsmöglichkeiten über Multi-Touch und dem eingebauten Beschleunigungssensor.

Die Entwicklung von eigenen Applikationen auf dem iPhone ist seit März 2008 möglich und ist für freie Entwickler besonders interessant, da Apple die nativen API's und Tools zur Verfügung stellt.

1.2 Aufgabenverteilung

Das Projekt kann im Wesentlichen in zwei Teile aufgeteilt werden. Die Implementierung der Augmented Reality Anwendung auf dem Server und die Entwicklung der iPod Touch Anwendung. Für die Implementierung der Augmented Reality Anwedung war Herr Henrik Brauer verantwortlich und für die Entwicklung der iPod Touch Anwendung war Herr Konrad Glugla verantwortlich.

Damit für die Bewertung unterschieden werden kann, wer welchen Teil verfasst hat, wurden die verschieden Teile mit KG für Konrad Glugla und HB für Henrik Brauer markiert (nur Abschnitte wurden markiert. Unterabschnitte gehören zum Autor des jeweiligen Abschnitts). Nicht markierte Abschnitte sind in Gemeinschaftsarbeit entstanden.

1.3 Aufbau der Arbeit

Diese Arbeit ist in fünf Kapitel eingeteilt. Das vorliegende Kapitel 1 gibt einen kurzen Überblick über die Motivation, Zielsetzung, die Aufgabenverteilung und den Aufbau der Arbeit.

Anschließend werden in Kapitel 2 einige Grundlagen erläutert, sowie die verwendeten Tools/ Frameworks vorgestellt.

Kapitel 3 beschreibt die Entwicklung des Systems. Dabei werden Konzepte und Implementierungen beschrieben sowie aufgetretene Probleme erläutert.

Kapitel 4 stellt die Anwendung vor, die für den Projekttag entwickelt wurde.

Das abschließende Kapitel 5 gibt eine Bewertung des Projektergebnisses, ein Fazit, sowie einen Ausblick auf weiterführende Arbeiten.

2 Grundlagen

2.1 ARToolkit (HB)

ARToolKit ist eine Softwarebibliothek zur Ermittlung der Lagedaten von grafischen Markern (auch Pattern oder Fiducials genannt) in einem per Kamera aufgenommenen Abbild der Realität. Die Lagedaten werden in Form einer 4x4 Matrix geliefert und bestimmen die Position der Marker relativ zur Kamera. Die Grafikausgabe erfolgt eigentlich mit OpenGL, diese wurde aber im Projekt durch eine 3D - Grafikengine ersetzt (siehe hierzu Abschitt 3) .

Abbildung 2.1: Marker für die Augmented Reality

Die Abbildung oben zeigt ein typisches ARToolKit-Pattern. Der dicke schwarze Rahmen dient zur Erkennung des Patterns und zur Bestimmung der meisten Lagedaten. Das weisse Feld (mit dem Muster) darin dient der Identifizierung des Patterns und gibt Aufschluss über die Ausrichtung (vier Möglichkeiten innerhalb des Rahmens). Um möglichst viele verschiedene Pattern zur Verfügung zu haben, entwickelten Forscher am Augmented Environments Laboratory (AEL) ein einfaches System zur Patternerzeugung. Das Schachbrettmuster am unteren Rand dient dabei der Ausrichtung des Patterns und das 3x4 große Feld darüber wird binär kodiert und ermöglicht somit 4096 Kombinationen.

ARToolKit ist nicht die einzige Möglichkeit um videobasierte Augmented Reality zu betreiben. ARToolKit ist jedoch ausgesprochen beliebt, da es keine spezielle (und damit teure) Hardware voraussetzt und relativ einfach zu verwenden ist. Zu erwähnen ist noch, dass es neben ARToolKit noch andere Softwarebibliotheken (z.B. ARTag) gibt, die die gleiche Funktionalität erfüllen. Andere Softwarebibliotheken wurden allerdings aus Zeitgründen nicht ausführlich getestet.

2.1.1 Gründe das ARToolkit zu benutzen

Das ARToolkit hat gegenüber anderen Tracking-Lösungen einige Vorteile, die es besonders für den Einstieg in die Augmented-Reality-Entwicklung geeignet erscheinen lassen.

Zunächst einmal ist das ARToolkit für nahezu alle relevanten Plattformen verfügbar. Das bedeutet, dass vorhandene Hardware eingesetzt werden kann anstatt sich nach den Anforderungen des verwendeten Tracking-Systems richten zu müssen. Ein anderer Vorteil des ARToolkit besteht darin, dass es im Quellcode verfügbar ist. Das erlaubt es zum einen, die genaue Vorgehensweise des Systems zu verstehen, zum anderen ermöglicht es die Behebung von etwaigen Fehlern, und zum dritten kann man Features, die man benötigt einfach selbst einbauen, ohne von anderen abhängig zu sein. Auch die Hardwareanforderungen des ARToolkits sind sehr moderat: Als Minimum wird eine 500Mhz CPU gefordert. Eine modifizierte Version des ARToolkits ist allerdings sogar auf einem PDA lauffähig. Im Projekt hat sich allerdings gezeigt das weit schnellere Hardware genutz werden musste. Dies kann allerdings auch am Zusammenspiel mit den anderen Komponenten gelegen haben. Die weiteren Hardwareanforderungen beschränken sich auf eine USB- oder Firewire-Webcam sowie einen Drucker um die Marker auszudrucken. Insgesamt kann man also sagen, dass das ARToolkit es mit seinen geringen Anforderungen auch weniger gut ausgestatteten Projekten erlaubt, Augmented Reality Anwendungen zu entwickeln.

2.1.2 Technische Details des ARToolkits

Die folgenden Abschnitte zur technischen Funktionsweise stellen viele der Vorgänge stark verkürzt dar. Für tiefergehende Beschreibungen sei auf den verfügbaren Quellcode des ARToolkits verwiesen.

Überblick über die Funktionsweise

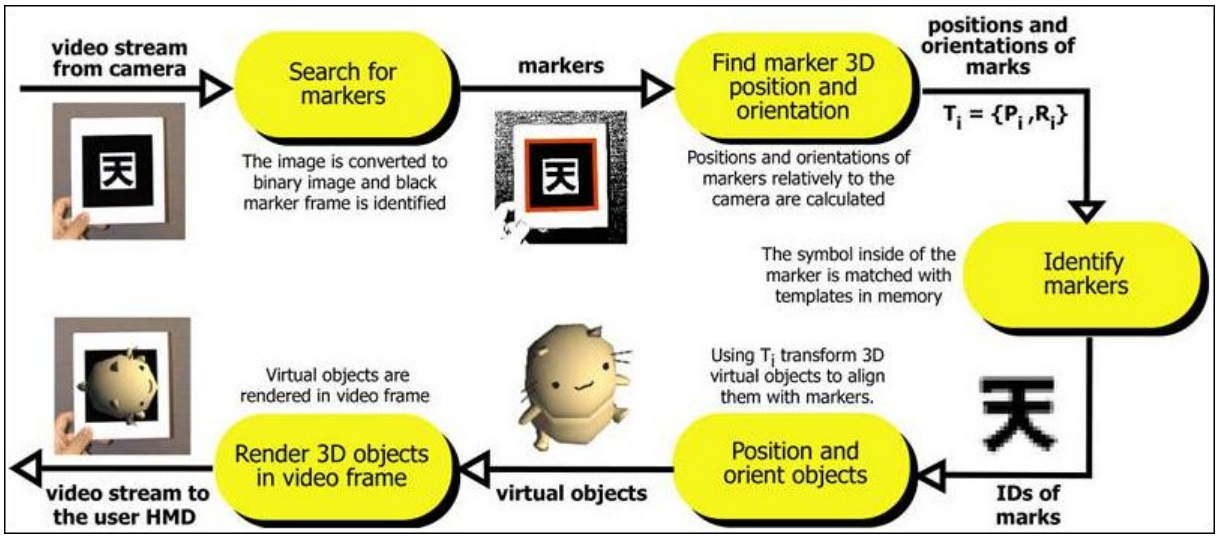

Abbildung 2.2: Überblick über die Funktionsweise des ARToolkits (Quelle: [ART09b])

Am Anfang steht ein Strom von Videodaten, der von der Kamera in das System hineinkommt. Die erste Aufgabe des ARToolkits ist es, in diesem Video-Stream nach Markern zu suchen. Der Inhalt des Markers spielt hier noch keine Rolle, lediglich die Form des schwarzen Rahmens wird in einem Schwellwert-gefilterten Binärbild gesucht.

Der nächste Schritt besteht darin, die 3D-Rotationen und Translationen zu errechnen, die die Position des Markers relativ zur Position der Kamera bestimmen. Die Funktionsweise dieses Teilschritts wird weiter unten („Die verschiedenen Koordinatensysteme“) etwas genauer beschrieben. Schließlich muss noch herausgefunden werden, um welchen Marker es sich überhaupt handelt. Dazu wird das Muster des aufgenommenen Markers in einem Pattern-Matching Verfahren mit den vordefinierten Markern verglichen und der Marker somit identifiziert.

Das jeweils zum Marker passende 3D-Objekt wird mit den Rotationen und Translationen des Markers relativ zur Kamera versehen. Das 3D-Objekt befindet sich somit an gleicher Position wie der Marker (genau so ist es auch möglich das 3D-Objekt relativ zum Marker darzustellen, z.B. 100 Pixel links vom Marker). Jetzt kann das 3D-Objekt gerendert und dargestellt werden. Wenn die beschriebenen Schritte für jeden Frame des Video-Streams durchlaufen werden entsteht für den Benutzer der Eindruck, dass das 3D-Objekt auf dem Marker „festgewachsen“ ist.

Die verschiedenen Koordinatensysteme

Wie in Abbildung 2.3 zu sehen ist, sind bei der Verwendung des ARToolkits mehrere verschiedene Koordinatensysteme im Spiel. Die eigentliche Aufgabe des ARToolkits ist es, eine Transformation vom Marker-Koordinatensystem (in der Abbildung blau) in das Kamera-Koordinatensystem (in der Abbildung schwarz) zu finden. Das Problem ist dabei, dass dem Video-Stream lediglich das verzerrte zweidimensionale Bildschirmkoordinatensystem zu entnehmen ist (in der Abbildung rot). Dieses muss mit Hilfe der Kameraparameter, die bei der Kamerakalibrierung ermittelt wurden, zunächst in das idealisierte Bildschirmkoordinatensystem (in der Abbildung grün) transformiert werden. Der nächste Schritt, aus den zweidimensionalen idealisierten Bildschirmkoordinaten die dreidimensionale Beziehung zwischen Kamera-Koordinatensystem und Marker-Koordinatensystem zu errechnen, ist der wohl algorithmisch anspruchsvollste Teil des ARToolkits. Die Aufgabe wird mit Hilfe eines iterativen Optimierungsprozesses gelöst. Die Startbedingungen sind die vier Ecken des Markers sowie Informationen aus dem vorhergehenden Frame des Video-Streams.

2.2 Irrlicht (HB)

Die Irrlicht Engine ist eine Open-Source Echtzeit 3D-Engine, welche sehr hohe Performance bietet. Geschrieben und verwendbar ist sie in C++. Für .NET-Sprachen, wie z.B. C#, ist sie auch verfügbar. Irrlicht ist plattformunabhängig und läuft unter anderem unter Linux, Mac OS, Sun Solaris und diversen Windows-Versionen.

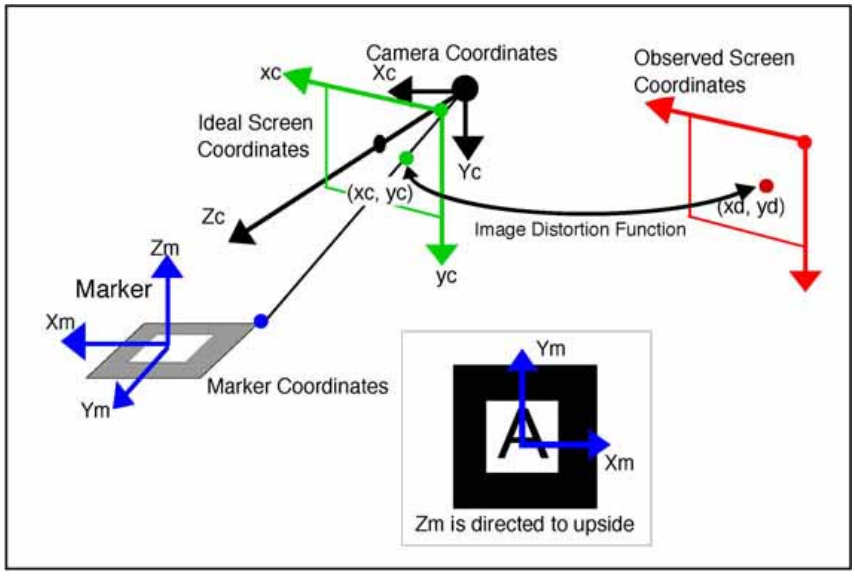

Abbildung 2.3: Koordinatensysteme des ARToolkits (Quelle: [ART09b]

Zu Irrlicht gibt es große Communities in mehreren Sprachen, sowie viele Projekte bei denen Irrlicht eingesetzt wird. Es gibt sehr viele Informationen und Material für Irrlicht, wie etwa Tutorials und Editoren, die die Entwicklung unterstützen.

Für die grafische Darstellung werden OpenGL, DirectX oder eines von zwei Programmen zum Rendern genutzt (Software Renderer). Ähnlich wie bei anderen Grafikengines wird ein Szenengraph verwendet. Irrlicht unterstützt den Entwickler von Vertex- oder Pixelshader durch einfache Integration selbiger.

Es existieren Klassen und Funktionen für Matrix- und Vektorrechnung und einfache Kollisionsabfrage. Der Quellcode ist offen in C++ geschrieben und steht unter der zlib-Lizenz, die auch eine Verwendung in kommerziellen Produkten ohne Herausgabe des Quelltextes ermöglicht.

Viele bekannte Dateiformate werden von Irrlicht unterstützt und können direkt von Irrlicht eingelesen werden. Dies gibt einem die Möglichkeit verschiedene Dateien ohne Konvertierung zu nutzen.

Irrlicht wird im Projekt für die Darstellung des Kamerabildes sowie für die Darstellung und die Transformation der 3D Objekte genutzt (sieh hierzu Abschnitt 3.1).

2.3 Multi-Touch (KG)

Unter Multi-Touch versteht man das Bedienen eines Computers oder eines Elektronischen Geräts über einen Touch-Screen (also einem berührungsempfindlichen Bildschirm) mit dem Einsatz von mehr als einem Finger.

Das 2007 erschienene Apple iPhone ist das erste Gerät für Privatanwender welches diese Technologie nutzt. Der Multi-Touch kommt hier z.B. beim Zoomen von Bildern zum Einsatz, bei der

der Anwender 2 Finger auseinanderziehen kann um das Bild zu vergrößern und die Finger wieder zusammenziehen muss um das Bild zu verkleinern.

2.4 Accelerometer (KG)

Der Accelerometer, oder auch Beschleunigungssensor, ist ein Sensor, der die Beschleunigung misst und somit Bewegungen erkennen kann. Dabei kann es sich um eine Positions- oder eine Winkelveränderung handeln.

Im Beschleunigungssensor arbeiten drei Elemente zusammen: Siliziummasse, Siliziumfedern und elektrischer Strom. Die Siliziumfedern messen mit Hilfe des elektrischen Stroms die Position der Siliziummasse (aus [APP09a]).

Durch den Beschleunigungssensor kann z.B. beim iPhone erkannt werden ob der Anwender das Gerät im Hoch- oder Querformat hält.

2.5 Applikationsentwicklung auf dem iPhone (KG)

Im März 2008 hat die Firma Apple die iPhone SDK vorgestellt und damit die Möglichkeit geschaffen eigene Applikationen für das iPhone zu entwickeln. Apple bietet die nativen API's und Tools an, mit denen auch alle von Apple entwickelten Applikationen erstellt wurden.

Das Betriebssystem des iPhone basiert auf Mac OS X und kann somit auf jahrelange Erfahrung zurückgreifen. In den Grundstrukturen ist das iPhone OS sehr ähnlich aufgebaut, die größte Abweichung findet sich im User Interface Application Framework, also dem Framework für den In- und Output. Der Input geschieht hier nicht über Tastatur und Maus, sondern über einen Bildschirm mit Multi-Touch.

Die Entwicklung von iPhone Applikationen kann mit den Tools von Apple nur auf Rechnern durchgeführt werden, auf denen das Betriebssystem Max OS X läuft.

Getestet werden können eigene Applikationen direkt auf dem iPhone, jedoch erst nach Anmeldung im kostenpflichtigen iPhone Developers Program, oder auf einem Simulator der in den Entwicklungstools bereitgestellt wird. Auf dem Simulator gibt es jedoch die Einschränkung, dass die Telefonfunktion, die Kamera und der Accelerometer nicht simuliert werden können. Außerdem wird nicht die Leistungsfähigkeit des iPhones simuliert, statt dessen nutzt der Simulator die vollen technischen Ressourcen des simulierenden Rechners. Daher sollte beachtet werden, dass die Performance auf dem Simulator nicht mit der auf dem iPhone zu vergleichen ist.

2.5.1 Objective-C

Die Applikationsentwicklung auf dem iPhone geschieht in der Programmiersprache Ojective-C. Objective-C erweitert die Standard ANSI C Bibliotheken, arbeitet aber objektorientiert. Die Syntax basiert größtenteils auf der Programmiersprache Smalltalk, einer der ersten objektorientierten Programmiersprachen.

Objective-C wurde unter der Leitung von Brad Cox in der 80er Jahren entwickelt und diente später als Basis für das Betriebssystem NextStep der Firma NeXT. NeXT wurde 1985 von Steve Jobs, dem jetzigen Geschäftsführer von Apple, gegründet und später von Apple komplett übernommen. Aufgrund dieser Entwicklung tragen viele Klassen der Objective-C Bibliothek das Kürzel „NS“ vor dem Klassennamen.

2.5.2 Xcode

Xcode ist die Entwicklungsumgebung von Apple. In Xcode ist nicht nur die Entwicklung von Objective-C Code möglich sondern auch Programmiersprachen wie z.B. Java, Python oder Ruby werden unterstützt. Xcode ist ein sehr mächtiges Tool welche das Coding unterstützt und eine umfangreiche Dokumentation bereit hält.

Neben dem Coding sind aus dieser Umgebung heraus alle weiteren Tools der iPhone Applikationsentwicklung erreichbar, wie z.B. der Interface Builder oder Instruments.

2.5.3 Interface Builder

In dem Interface Builder wird das grafische User Interface für die iPhone Applikation visuell gebaut. Vordefinierte Komponenten, wie z.B. Buttons und Textfelder, können im Interface Builder per drag and drop in die GUI gezogen und positioniert werden. Damit der Code mit den Objekten der GUI arbeiten kann, werden über den Interface Builder Verbindungen zwischen den Objekten und dem Applikationscode hergestellt.

Die GUI kann zwar auch durch Programmcode erstellt werden, jedoch wird das Erstellen über diesen grafischen Editor deutlich erleichtert, vor allem da das Ergebnis direkt sichtbar ist.

2.5.4 Instruments

Mit Instruments lässt sich die Performance von laufenden iPhone Applikationen, auf dem Simulator oder dem Endgerät, analysieren. Instruments sammelt Daten, wie z.B. Speichernutzung, Netzwerkaktivität und grafische Performance, der laufenden Applikation, und stellt sie in einer grafischen Übersicht dar. Auf einer Zeitachse kann der Verlauf der Werte beobachtet werden.

Über Instruments lassen sich auch Speicherlecks erschliessen. Alle Methodenaufrufe können detailliert beobachtet und nachverfolgt werden.

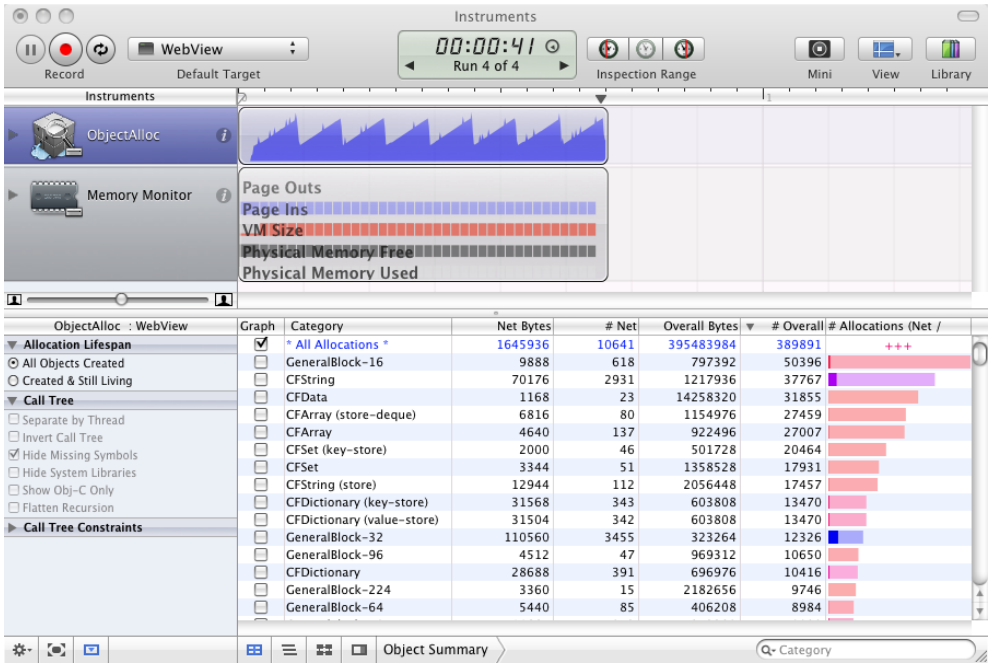

Abbildung 2.4: Performance Check in Instruments

Abbildung 2.4 zeigt beispielhaft den Instruments Bildschirm einer stabilen Applikation.

3 Umsetzung

Dieses Kapitel beschreibt die Implementierung der verschiedenen Komponenten die im Projekt entwickelt wurden. Hierbei wird auf die Idee, das Vorgehen und die Probleme eingegangen.

Die Abbildung 3.1 gibt einen Überblick über die implementierten Komponenten.

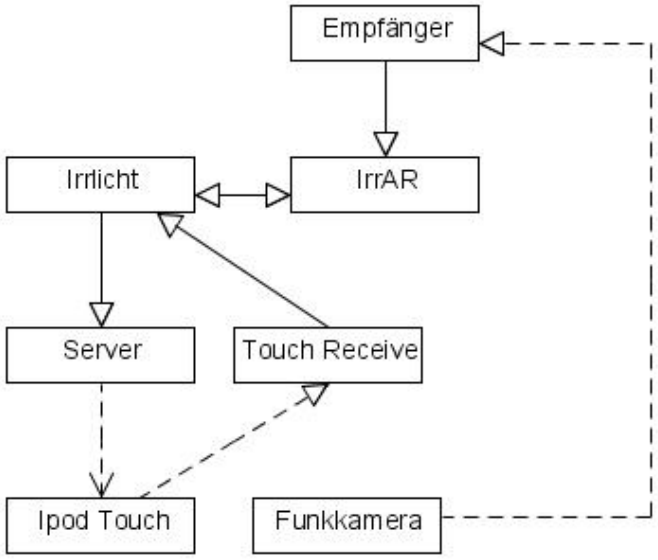

Abbildung 3.1: Überblick über die implementierten Komponenten

3.1 Irrlicht (HB)

Eine in Irrlicht erstellte Anwendung besteht im Wesentlichen aus 2 Teilen.

1. Erzeugung der 3D Objekte (in Irrlicht Nodes genannt).
2. Die Hauptschleife.

Die Nodes werden in der Regel vor Beginn des Renderns erzeugt. Üblicherweise werden für die Erzeugung der Nodes Polygonnetze aus einer Datei geladen. Allerdings bietet Irrlicht auch mehrere vorgefertigte Grundelemente, wie z.B. einen Würfel, die genutzt werden können. Zusätzlich besteht noch die Möglichkeit ein Polygonnetz dynamisch im Code zu erzeugen. Nach

dem Erzeugen der Node kann ihr noch eine Textur zugewiesen und eine Position festgelegt werden.

Nach dem Erzeugen aller Nodes beginnt die Hauptschleife. Die Hauptschleife ist im Wesentlichen eine Endlosschleife in der jedes mal die aktuelle Position der Nodes berechnet und dann neu gezeichnet wird.

3.2 ARToolKit (HB)

Um mit Hilfe von Irrlicht Augmented Reality zu erzeugen muss neben Irrlicht zusätzlich das ARToolKit genutzt werden.

Wie schon in Teil 2.1 beschrieben bestimmt das ARToolKit die Position der Marker relativ zur aktuellen Kameraposition. An diese Position wird dann mittels Irrlicht das passende 3D Objekt gezeichnet. Hierfür wird wie folgt vorgegangen. Als erstes wird mit Hilfe des ARToolKits das aktuelle Kamerabild geladen. Dieses Bild wird dann wie in Teil 2.1.2 schon beschrieben nach verschieden Markern durchsucht. Falls ein Marker gefunden wurde, wird die Rotationsmatrix, die vom ARToolKit erzeugt wird, angepasst und an den entsprechenden Irrlicht Node weitergeleitet. Falls der passende Marker, der an eine Node gekoppelt ist, nicht sichtbar ist wird diese Node auf „nicht sichtbar“ gesetzt.

Um diese Funktionalität möglichst einfach nutzen zu können wurde eine Klasse auf Basis von [IRR08] entwickelt die einen leichten Umgang mit den einzelnen Komponenten bietet. Über diese Klasse wird jeder Node einem Marker zugeordnet. Es ist auch möglich einer Node mehrere Marker zuzuweisen, sogenannte Multi-Marker. Diese haben den Vorteil, dass eine höhere Erkennungsrate vorhanden ist. Also falls einer der Marker nicht sichtbar ist kann ein anderer genutzt werden.

Neben der Zuordnung der Marker ist die Klasse auch noch für das Umwandeln des Kamerabilds in eine Irrlicht Textur verantwortlich. Jegliche Kommunikation mit dem ARToolKit wird in dieser Klasse gekapselt.

3.2.1 Problem: Transformation bei Bewegung

Die 3D Welt ist so konzipiert, dass die Kamera eine feste Position hat. Alle Objekte werden entsprechend der Markerposition im Weltkoordinatensystem verschoben. Dieser Ansatz hat den Vorteil das vorher nicht bekannt sein muss wie die verschiedenen Marker im Verhältnis zueinander im Raum verteilt sind. Hat aber den Nachteil, dass einfache Animationen, wie z.B. die Bewegung auf einer Achse, durch die Rotation und Verschiebung des Objektes komplexer werden. Beispielsweise sollte eine Figur auf der x-Achse von links nach rechts laufen. Durch die Rotation lief die Figur dann zwar noch von links nach rechts aber zusätzlich noch im Bild nach oben, was dazu führte dass die Figur größer wurde, was natürlich nicht beabsichtigt war. Anfänglich wurde

versucht Lösungen zu finden um trotzdem möglichst einfach Animationen darzustellen. Aus zeitlichen Gründen wurde entschieden dass die Animationen auf Rotationen um die eigene Achse beschränkt werden, da dabei die oben genannten Probleme nicht zum Tragen kommen.

3.3 Kommunikation Server (HB)

Wie im Abschnitt 3.1 schon beschrieben findet die Anzeige der Bilddaten mit Hilfe von Irrlicht statt. Irrlicht ist allerdings nur dafür ausgelegt die Bilddaten direkt auf dem Bildschirm darzustellen. Irrlicht bietet von Haus aus keine Möglichkeit die Bilddaten zu streamen. Aus dem Grunde musste eine andere Möglichkeit gefunden werden. Eine wichtige Vorrausetzung war, dass die Daten in Echtzeit auf dem Client dargestellt werden bzw. mit so geringem Jitter, dass dieser vom Betrachter nicht merklich wahr genommen wird.

Die erste Idee hierbei war es aus den Bildinformationen einen permanenten Videostream zu erzeugen und diesen dann zu broadcasten. Es wurde aber keine Möglichkeit gefunden einen Videostream auf einfache Art und Weise selbst zu erzeugen.

Deshalb war der zweite Ansatz ein Tool zu nutzen, dass den Bildschirm „abfilmt“ und die Daten dann broadcastet. Ein Tool dass diese Aufgabe direkt übernehmen kann wurde allerdings nicht gefunden, was dazu führte dass zwei Tools genutzt werden mussten. Eines dass den Bildschirm „abfilmt“ und sich so in das System einbindet, so dass das Bildschirmbild wie das Signal einer Webcam genutzt werden kann. Sowie ein zweites Tool, dass die Daten einer Webcam (hier unser Bildschirm Bild das als Webcam dargestellt wird) als Live-Stream zur Verfügung stellt.

Dieses Tool bietet die Möglichkeit 3 verschiedene Arten von Streams zu erzeugen. Bei der einfachsten Variante werden JPG Bilder erzeugt und auf dem Zielsystem mit Hilfe eines Java Scripts nachgeladen. Bei der zweiten Art wird ein Flashplayer zur Darstellung genutzt und beim dritten Ansatz ein Java Applet. Da die Nutzung von Flashplayer und Java Applet auf dem iPod nicht ohne weiteres möglich ist, wurde die einfache Variante mit dem Java Script genutzt.

Dieser Ansatz hat im Prinzip sehr gut funktioniert allerdings führte er zu Performance Problemen. Um dieses Problem zu umgehen wurde das Erzeugen der JPG und das zur Verfügung stellen der erzeugten JPG mit Hilfe eines Webservers selbst implementiert.

Hierfür musste als erstes die Irrlicht Engine angepasst werden. Irrlicht bot in der aktuellen Version (1.4.2) zwar die Möglichkeit Screenshots zu erzeugen und dies als JPG auf der Festplatte zu speichern. Hatte aber nicht die Möglichkeit die Datei virtuell im Arbeitsspeicher zu erzeugen um sie dann direkt aus dem Programm heraus über das Webserver Modul auszugeben. Auf Nachfrage im Irrlicht Forum wurde diese Funktionalität nachträglich implementiert und konnte für unsere Anwendung genutzt werden.

Im zweiten Schritt musste ein Webserver selbst implementiert werden. Dieser war nötig weil das Nutzen eines externen Webserves wie z.B. Apache Tomcat die Echtzeitanforderungen auf dem Zielsystem nicht mehr gewährleistet hätte. Um den Webserver nicht komplett selbst implementieren zu müssen wurde ein Beispiel Webserver ([WEB09]) genommen und entsprechend der Anforderungen angepasst. Das Hauptprogramm und der Server laufen in unterschiedlichen

Threads, zusätzlich bekommt noch jeder Client einen eigenen Thread zugewiesen. Das gewährleistet, dass die verschieden Programmteile nicht auf Grund von Wartezeiten in der Kommunikation ausgebremst werden und durchgehender Echtzeitbetrieb möglich ist.

Die Kommunikation zwischen dem Augmented Reality Teil des Programms und dem Server läuft über ein einfaches Daten Array. In dieses Array schreibt Irrlicht jeweils das neu erzeugte Bild und der Server gibt dieses Bild, bei Anfrage nach einem neuen Bild, aus.

3.4 Anzeige und Interaktion (KG)

Als mobiles Endgerät zur Darstellung der Augmented Reality stand uns ein iPod Touch der 2. Generation zur Verfügung. Bis auf einige Einschränkungen in der Ausstattung, wie z.B. der Kamera und der Telefonfunktion, hat es die selben Eigenschaften wie das iPhone und kann auch mit den nativen Tools und API's von Apple zur Entwicklung eigener Applikationen eingesetzt werden.

Auf dem iPod Touch sind 2 Komponenten zu implementieren:

1. Die Anzeige des Video-Streams der Augmented Reality Anwendung
2. Die Verarbeitung der Touch Interaktion

3.4.1 Anzeige des Video-Streams

Wie bereits erwähnt wird das Bild der Augmented Reality Anwendung über einen Webserver zur Verfügung gestellt.

Der iPod Touch verfügt über einen integrierten Webbrowser(Safari) über den das Bild vom Webserver angezeigt werden kann. Dieser Webbrowser kann in eigenen Applikationen eingesetzt werden indem das Modul „UIWebView“ implementiert wird. Zur Anzeige über diese Schnittstelle wurde bei der Umsetzung aber abgesehen, da es hierbei zu starken Performance Problemen gekommen ist (mehr dazu in Abschnitt 3.6).

Zur Anzeige der Augmented Reality Umgebung müssen aktuelle Bilder in einer ausreichenden Geschwindigkeit nachgeladen werden. Um für das menschliche Auge eine flüssige Bewegung zu simulieren reichen in der Regel 25 Bilder pro Sekunde.

Das User Interface besteht lediglich aus der Bilddarstellung. Daher wurde über den Interface Builder lediglich das Modul „UIImageView“ eingebunden und deckt das komplette Display ab (480x320 Pixel).

Das Laden der Bilder geschieht in einem einfachen Thread, der alle 40ms ein neues Bild vom Server nachlädt. Durch das Neuladen der Bilder in einem Abstand von 40ms erreichen wir die 25 Bilder pro Sekunde. Um den Speicher und die Rechenleistung nicht unnötig zu beanspruchen wird das Laden der Bilder im „Unchached“ Modus durchgeführt. So kann das Zwischenspeichern von geladenen Bildern verhindert werden.

Performance Optimierung

Während der Implementierungsphase kam es zu deutlichen Performance Problemen. Da die Performance bei dieser Echtzeitanwendung aber eine große Rolle spielt, damit die Augmented Reality für den Anwender auch ein Erlebnis ist, wurde auf die Performance Optimierung ein besonderes Augenmerk gelegt.

Bei der Performance Optimierung wurde das in Xcode integrierte Tool „Instruments“ eingesetzt, in welchem Speichernutzung, Objektallokierung etc. während der Applikationsausführung überwacht werden können. Mit Hilfe dieses Tools konnten einige Codestellen optimiert werden, wodurch eine kleine Performancesteigerung erzielt werden konnte. Als Beispiel sei genannt, dass das Laden von Bildern mit verschiedenen Methoden und auswählbaren Optionen durchgeführt werden kann.

Da das Zuweisen des empfangenen Bildes auf das Display eine gewisse Zeit in Anspruch nimmt wurde eine Alternativlösung mit 2 Threads implementiert, bei der ein Thread sich nur um das Empfangen der Bilder kümmert und ein Thread um die Darstellung auf dem Display.

Durch zusätzliche Sichtproben am Endgerät brachte diese Lösung die beste Performance und wurde für das Endprodukt eingesetzt. Das Konzept dieser Lösung ist in Abbildung 3.2 illustriert.

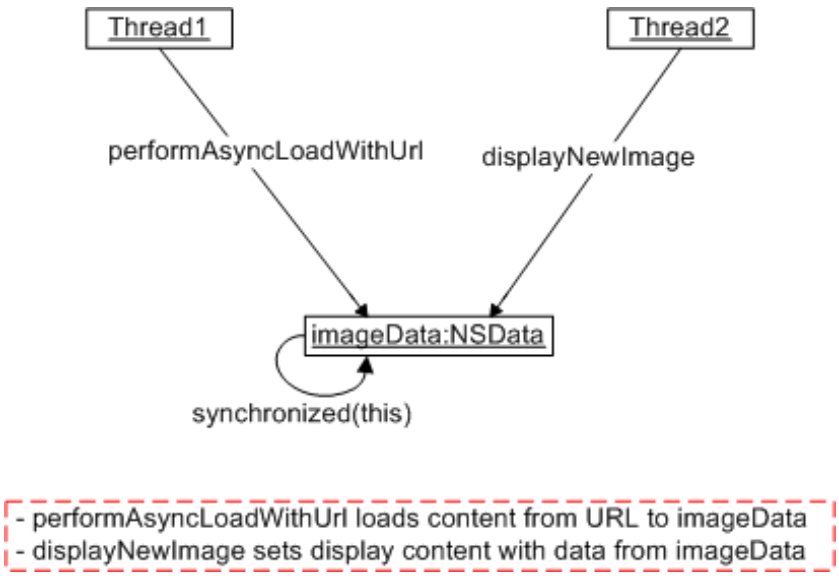

Abbildung 3.2: Kollaborationsdiagramm zur Lösung der Bilddarstellung mit 2 Threads

3.4.2 Touch Interaktion

Der Anwender sollte zusätzlich zur Anzeige der Augmented Reality die Möglichkeit haben, mit der Augmented Reality zu interagieren. Für die Interaktion wurde die Multi-Touch Funktionalität des iPod Touch eingesetzt. Implementiert werden musste hier die Gestenerkennung sowie die

Kommunikation mit der Augmented Reality Anwendung, um die erkannten Gesten zu verarbeiten.

Folgende potenziell durchführbare Gesten wurden für unsere Anwendung identifiziert:

- Drehen von Objekten in x- und y-Richtung durch streichen über den Bildschirm (swipe)
- Verkleinern und vergrößern von Objekten (Zoom) durch das Auseinander- oder Zusammenziehen von 2 Fingern
- Anhalten von bewegenden/ drehenden Objekten durch einfachen Fingertipp
- weitere ausführbare Kommandos durch doppelten Fingertipp, z.B. um Figuren an eine bestimmte Position zu schicken

Das Implementieren des Zoomens von Objekten wurde für die erste Umsetzung vernachlässigt, da die Augmented Reality Anwendung das Verkleinern und Vergrößern von Objekten beinhaltet, indem man sich dem Marker nähert bzw. entfernt.

Die Idee für das Drehen von Objekten war es, einen 3D-Würfel, der auf jeder Seite verschiedene Bilder oder Texte beinhaltet, durch Fingerbewegung in x- und y-Richtung drehen zu lassen. Letztendlich haben wir die Möglichkeit der Drehung eingeschränkt, indem wir nur Drehungen um die y-Achse (Drehung des Würfels nach links oder rechts) zugelassen haben. Diese Entscheidung wurde getroffen, da durch das Drehen in x- und y-Richtung sich der Würfel für den Anwender zu unkontrolliert dreht und er somit schwer wieder in eine Position gebracht werden kann, bei der die Bilder und Texte auf dem Würfel gut erkennbar sind.

Für die Geste „touchesMoved“, also das Bewegen des Fingers auf dem Display, wird bereits eine Methode bereitgestellt. Implementiert werden musste bei dieser Geste also lediglich das Erkennen der Bewegungsrichtung sowie die Geschwindigkeit der Bewegung. Die Bewegungsrichtung wird durch Vergleich von nacheinander folgenden Positionsdaten berechnet. Zur Berechnung der Geschwindigkeit wird die Zeit gestoppt die der Finger für das Zurücklegen einer bestimmten Strecke benötigt. Kurz zusammengefasst ergeben sich 2 implementierte Methoden:

- **analyseMovementWithX**
  Berechnet die Bewegungsrichtung durch Vergleich mit vorigen Positionsdaten
- **calculateSpeedWithDistance**
  Berechnet die Geschwindigkeit der Bewegung anhand von zurückgelegter Distanz und Zeitdifferenz

Die nächste Geste ist ein einfacher Fingertipp auf dem Bildschirm, wodurch ein Stopp-Signal ausgelöst werden soll, dessen Implementierung trivial ist. Weitere Gesten wurden mangels Zeit in diesem Projekt nicht mehr realisiert.

Nach Interpretation der Gesten müssen diese nun an die Augmented Reality Anwendung durchgereicht werden. Hierfür wird über WLAN eine UDP-Verbindung mit der Augmented Reality Anwendung aufgebaut und über diesen Kanal die interpretierten Gesten als Signale (Strings) gesendet. Die Anwendung umfasst folgende Signale:

- „*SwipeX %f*“ (%f wird durch eine float-Zahl ersetzt und beschreibt die Geschwindigkeit)
- „*SwipeY %f*“ (in der Endversion nicht mehr vorhanden)

- „*STOP*“

3.5 3D Objekte (KG)

Als letzten Schritt wurden 3D Objekte benötigt die in der Augmented Reality Anwendung dargestellt werden sollten. Zur Erstellung von 3D Objekten wurde das Freeware Tool Blender eingesetzt. Die Projektidee war 3D Würfel zu erstellen, auf denen Bilder und Texte den Entwicklungsprozess der Teilprojekte im „Emotional Tent“ illustrieren.

Zusätzlich entstand die Idee von animierten Figuren, welche die Teilprojekte virtuell erweitern sollten. Mit diesen Figuren sollte per Fingertipp interagiert werden. Auf Grund der hohen Einarbeitungszeit in Blender und den Performance Problemen bei der Anzeige des Videobildes auf dem iPod Touch (siehe Abschnitt 3.6), konnte dieser Punkt bis Projektende leider nicht umgesetzt werden.

Der 3D Würfel bekommt auf jede Seitenfläche eine Textur die aus einem Bild besteht. Die Kamera- und Lichteinstellungen müssen hier nicht berücksichtigt werden, da diese in der Irrlicht Anwendung eingestellt werden. Das Objekt kann direkt aus Blender exportiert und von der Irrlicht Anwendung geladen werden. Abbildung 3.3 zeigt beispielhaft einen in Blender gerenderten Würfel.

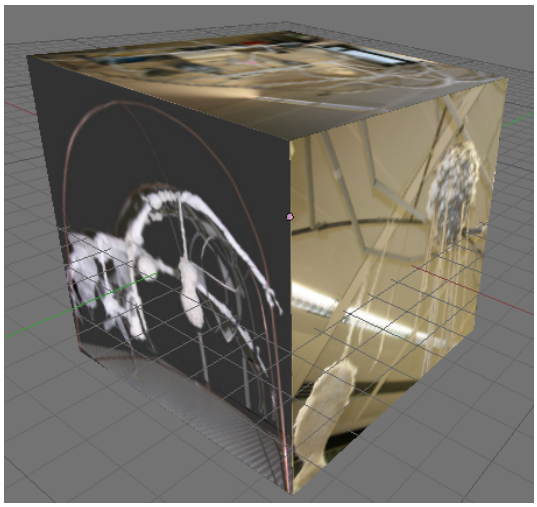

Abbildung 3.3: 3D Würfel mit Textur (Screenshot aus Blender)

3.6 Probleme bei der Umsetzung (KG)

Bei der Umsetzung auf dem iPod Touch kam es kurz vor Projektende noch zu einigen Performance Problemen. Die erste Umsetzung zur Anzeige des Video-Streams über das Modul „UIWebView“, mit der HTML Seiten dargestellt werden können, schien optimal zu laufen. In einer

etwas geringeren Auflösung waren auch keine Einschränkungen bei der Performance zu erkennen. Skalierte man die Anzeige jedoch über den kompletten Bildschirm, war die Videoanzeige nur noch sehr ruckelhaft. Viel Zeit wurde hier verloren um die Fehlerquelle zu finden und zu beheben. Auch mehrere E-mails an den Apple Support konnten hier keine Lösung bringen.

So musste nach einer Alternativlösung gesucht werden, welche darin bestand einzelne Bilder in einer ausreichenden Geschwindigkeit nachzuladen. Diese Lösung hatte die groben Performance Probleme vorerst behoben, zumindest auf dem Simulator. Wurde die Applikation auf das Gerät gebracht lief die Anzeige erneut nicht zufriedenstellend und führte sogar gelegentlich zu Abstürzen.

Auf der Suche nach der Fehlerquelle brachte auch das Tool „Instruments“ keinen Hinweis auf Fehler bei der Programmierung. Es war uns unerklärlich wodurch die Fehler auftraten, zumal es auf dem Simulator einwandfrei lief. Erst nach etlichen Testläufen konnte festgestellt werden, dass der Fehler durch die Konstruktion mit der Funkkamera aufgetreten war. Die Funkwellen der Kamera haben das WLAN Signal des iPod Touch gestört und brachte somit Probleme beim Empfang der Bilder über WLAN. Entfernte man die Funkkamera vom iPod Touch, so lief die Anwendung einwandfrei.

Das Problem mit der Funkkamera hat viel kostbare Projektzeit beansprucht. Vor allem die Arbeit in Blender musste auf Grund der Performance Probleme unterbrochen werden, da ein flüssiges Anzeigebild bei dieser Echtzeitanwendung höchste Priorität hatte.

4 Augality

Augality ist eine Anwendung die die Besucher über den Entwicklungsprozess des „Ambient Awareness“ Projekts informiert hat. Hierbei war ein besonderes Ziel ein interaktives Erlebnis zu bieten. Der Besucher konnte mit Hilfe des Versuchsaufbaus, der aus iPod Touch sowie einer Funkkamera bestand, verschiedene Marker betrachten. Oberhalb und unterhalb der Marker wurden Zettel montiert, auf denen kurz erläutert wurde für welches Teilprojekt dieser Marker stand. Anstelle der Marker wurde in Augality ein 3D Würfel eingeblendet auf dem sich verschiedene Bilder, passend zu den Projekten, befanden. Dies waren weitestgehend Bilder vom Aufbau und der Entwicklung der einzelnen Projekte.

Die Darstellung fand auf dem iPod Touch statt. Neben der Möglichkeit sich die Würfel von verschiedenen Seiten anzuschauen, in dem man aus verschiedenen Perspektiven auf die Marker schaute, bestand noch die Möglichkeit die Würfel zu drehen. Hierfür hat man den Finger auf dem Touchdisplay des iPod Touchs entweder nach links oder rechts gezogen. Entsprechend der gewählten Richtung drehten sich die Würfel. Durch ein Tippen auf das Display konnte die Drehbewegung wieder gestoppt werden.

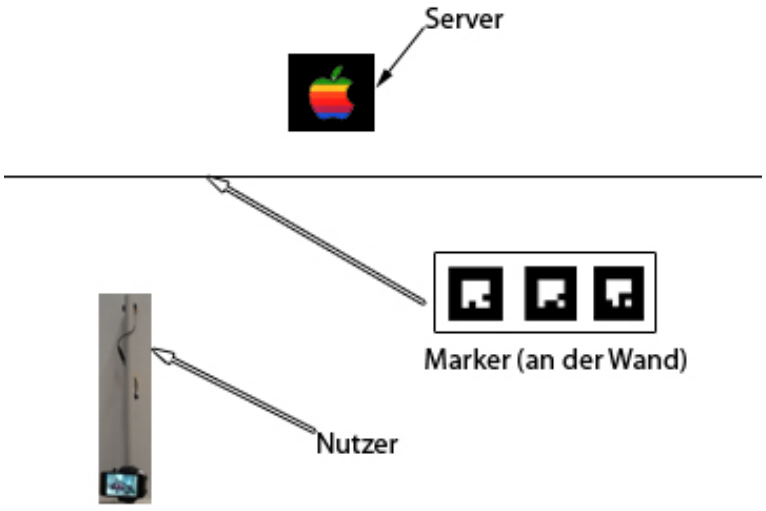

Abbildung 4.1: Versuchsaufbau der Augality Anwendung

Abbildung 4.2: Konstruktion der Funkkamera mit iPod Touch

In Abbildung 4.2 ist links die Konstruktion der Funkkamera mit dem iPod Touch in der ersten Version zu sehen. Auf Grund der störenden Funkwellen musste die Funkkamera vom iPod Touch getrennt werden (rechtes Bild).

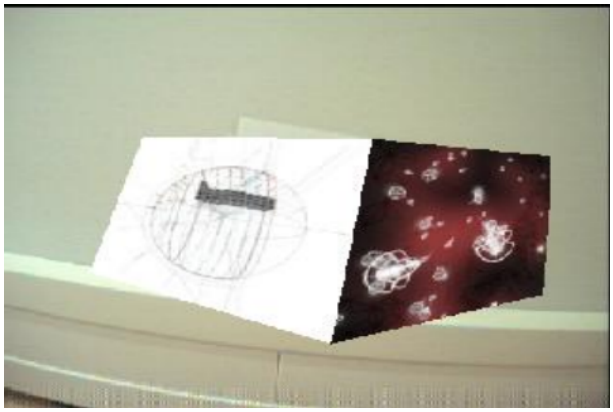

Abbildung 4.3: Screenshot der Augality Anwendung auf dem iPod Touch

5 Resümee

5.1 Zusammenfassung

Der vorliegende Projektbericht beschreibt die Entwicklung einer Augmented Reality Anwendung auf dem iPod Touch. Hierfür werden Bildaten einer Funkkamera, die am iPod Touch befestigt ist, auf einem Server ausgewertet und mit dem ARToolKit nach Markern durchsucht. Anstelle der Marker werden mit Hilfe von Irrlicht 3D Objekte eingeblendet. Das erzeugte Bild wird vom iPod Touch über WLAN vom Server geladen und angezeigt. Durch Berührung des Touchdisplays am iPod Touch ist es möglich mit den Objekten zu interagieren.

5.2 Fazit

Im Laufe des Projekts hat sich gezeigt, dass das ARToolKit ein sehr einfach zu nutzendes Tool ist das allerdings Schwächen in der Markererkennung zeigt. Vor allem bei schlechten Lichtverhältnissen wurden Marker oft nicht erkannt. Außerdem kam es häufig vor, dass Marker an Stellen erkannt wurden an denen sich keine Marker befanden. Dies hat gerade bei Nutzern die nicht mit der Funktionsweise des ARToolKits vertraut waren zu Irritationen geführt. Für spätere Projekte sollte deshalb nach Alternativen gesucht werden die eine bessere Erkennungsrate haben.

Die Bilder der Funkkamera waren sehr oft verrauscht, selbst wenn sich der Empfänger in unmittelbarer Nähe der Kamera befand. Sobald die Kamera mehr als 10 Meter vom Empfänger entfernt war, war das Bild so schlecht, dass es nicht mehr möglich war die Kamera zu nutzen. Zusätzlich gab es noch das in Abschitt 3.6 beschriebene Problem mit der Störung des iPod Touch. In einem zukünftigen Projekt sollte deshalb eine Alternative für die Funkkamera gefunden werden.

Der Einsatz eines iPod Touch als Anzeigegerät für Augmented Reality bewies sich im Projekt als zufriedenstellend, wobei der iPod Touch hier nur eingesetzt wurde um ein Videobild zu simulieren. Da die eigentliche Verarbeitung der Augmented Reality aber nicht direkt auf dem iPod Touch stattfand bleibt hier offen ob der iPod Touch genug Leistung bringt um Augmented Reality direkt auf dem Gerät zu berechnen. Internetrecherchen dazu zeigten dass es funktioniert, jedoch die Performance nur im akzeptablen Bereich liegt.

Der Einsatz des Multi-Touch Screens war vom Aufwand der Entwicklung sehr überschaubar und hat bei der Ausstellung für wenig Erklärungsbedarf gesorgt. Die Steuerung über Multi-Touch Screens ist sehr intuitiv und macht den meisten Anwendern Spaß.

Die Entwicklung auf dem iPod Touch war jedoch mit einigen Startschwierigkeiten verbunden, da die Einarbeitung in die eher selten eingesetzte Programmiersprache Objective-C sehr viel Zeit beansprucht hat. Vor allem die Syntax und die Arbeit ohne Garbage-Collection stellten eine große Hürde dar. Durch ausführliche Dokumentationen und das Vorhandensein vieler Tutorials (von Apple sowie von freien Entwicklern) konnte letztendlich die Umsetzung mit zufriedenstellendem Ergebnis erreicht werden. Die Entwicklungstools von Apple sind schon sehr ausgereift und unterstützen den Entwickler sehr gut bei der Arbeit. Allen voran die automatische Codevervollständigung und der direkte Zugriff auf die Dokumentation aus dem Quellcode heraus.

Blender ist ein sehr mächtiges Tool. In dem Projekt konnte jedoch nicht viel Zeit in die Arbeit mit Blender gesteckt werden. Die Einarbeitung in Blender kostet ziemlich viel Zeit doch nach den ersten Hürden ist die Arbeit in Blender auch mit Spaß verbunden. Die Animation von 3D Objekten über Blender wird auf Grund des großen Spektrums an Möglichkeiten und aus privatem Interesse an der 3D Modellierung höchstwahrscheinlich von beiden Projektteilnehmern weiter verfolgt.

Insgesamt kann gesagt werden, dass die Ziele trotz vieler Probleme erreicht wurden. Die Idee, die Bilderkennung auf dem Server durchzuführen und nur das Bild auf dem iPod Touch darzustellen, hat sehr gut funktioniert und kann in gleicher Weise auch für spätere Projekte genutzt werden.

5.3 Ausblick

Ein mögliches Szenario für den Einsatz von Augmented Reality auf dem iPhone wäre z.B. eine Art Museumsführer. Besucher mit iPhone könnten sich beispielsweise gratis die Museums-Applikation aus dem App-Store laden (ein Online erreichbarer Store in dem eigene Applikationen angeboten werden können) und mit dieser Applikation durch das Museum laufen, in der die einzelnen Ausstellungen mit Augmented Reality virtuell bereichert werden. Beispielsweise könnten alte Bauwerke in einer 3D-Animation in ihrem ursprünglichen Zustand angezeigt werden.

Literaturverzeichnis

[APP09a] *Apple Inc.* Webseite, Stand: 21. Februar 2009. – http://www.apple.com/iphone/features/accelerometer.html

[APP09b] *Apple Inc.: iPhone Dev Center.* Webseite, Stand: 21. Februar 2009. – http://developer.apple.com/iphone

[ART09a] *ARTag.* Webseite, Stand: 22. Februar 2009. – www.artag.net/

[ART09b] *ARToolKit.* Webseite, Stand: 22. Februar 2009. – www.hitl.washington.edu/artoolkit/

[Bic03] BICHLER, Simon: *ARToolkit.* Webseite, 1. Dezember 2003. – http://wwwnavab.in.tum.de/twiki/pub/Far/AugmentedRealityBenutzerschnittstellenWiSe2003/hauptseminar-artk.pdf

[ENC09] *The Free Online Encyclopedia.* Webseite, Stand: 21. Februar 2009. – http://encyclopedia2.thefreedictionary.com

[IRR08] *IrrAR.* Webseite, Stand: 18. Oktober 2008. – http://sourceforge.net/projects/irrar/

[IRR09] *Irrlicht 3D.* Webseite, Stand: 22. Februar 2009. – http://irrlicht.sourceforge.net/

[OSR09] *Operating System Reviews.* Webseite, Stand: 21. Februar 2009. – http://www.operating-system.org/betriebssystem/_german/bs-nextstep.htm

[TEN09] *Tenon Intersystems : Objective-C.* Webseite, Stand: 21. Februar 2009. – http://www.tenon.com/products/codebuilder/Objective-C.shtml

[WEB09] *Charlotte: A Simple Web Server for Microsoft Windows.* Webseite, Stand: 22. Februar 2009. – http://www.acm.org/crossroads/xrds6-4/charlotte.html

[WIK09a] *Wikipedia: Multi-Touch.* Webseite, Stand: 21. Februar 2009. – http://de.wikipedia.org/wiki/Multi-Touch

[WIK09b] *Wikipedia: Objective-C.* Webseite, Stand: 21. Februar 2009. – http://de.wikipedia.org/wiki/Objective-C