UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

InformationsZentrum Sozialwissenschaften

Schnelle Prototypenentwicklung für Augmented Reality

Studienarbeit

vorgelegt von

Jens Anhenn
(marvin@uni-koblenz.de)

Erstgutachter: Prof. Dr. Jürgen Krause
(Institut für Computervisualistik, AG Softwareergonomie)

Zweitgutachter: Philipp Schaer
(Institut für Computervisualistik, AG Softwareergonomie)

Koblenz, im Februar 2007

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden und stimme der Veröffentlichung dieser Arbeit im Internet zu.

_____

Ort, Datum Unterschrift

Inhalt

1 Einführung

In weitgehend allen Bereichen der Industrie ist die Einbindung von Prototypen in den laufenden Entwicklungsprozess schon seit langem Usus. Bekannt ist diese Vorgehensweise vorrangig aus der Automobilindustrie, wo mit zum Teil nicht unerheblichem Aufwand Testmodelle erstellt werden, um Design- oder Ergonomieprobleme genauer untersuchen zu können. In neuerer Zeit wird diese Aufgabe zunehmend unter anderem aus Kostengründen mit Hilfe von Computern gelöst. Virtuelle Modelle übernehmen hier die Aufgabe von früher per Hand gefertigten realen Modellen. Sowohl virtuelle Realität als auch *Augmented Reality* (hierbei werden virtuelle Objekte mit der Realität kombiniert, indem sie in das Sichtfeld des Betrachters eingeblendet werden) gewinnen dadurch stetig an Gewicht.
Aber auch abseits großer Entwicklungszentren ist die zunehmende Präsenz dieser Technologien zu beobachten.

Liegt das Augenmerk bei der Entwicklung derzeitiger AR-Systeme nach wie vor auf der technischen Umsetzung und Echtzeit-Fähigkeit, so ist doch mit steigender Funktionalität und somit auch Komplexität über eine komfortable wie auch intuitive Bedienbarkeit und Interaktion mit dem System nachzudenken. Ein entscheidender Aspekt bei der Entwicklung von benutzerfreundlicher Software ist die möglichst frühe Integration des Benutzers in den Entwicklungsprozess. Je früher dies erfolgt, desto mehr Erkenntnisse kann man aus der Testsituation gewinnen, desto schneller muss aber auch auf Veränderungen reagiert und ein geeigneter Prototyp entwickelt werden. Dies stellt sich bei AR-Systemen deutlich schwerer dar, weil im Gegensatz zu Desktop-Systemen in diesem Bereich nicht annähernd die Menge an Tools und Techniken zur Verfügung stehen.

Der Schwerpunkt dieser Arbeit soll auf der schnellen sowie einfachen Umsetzung eigener Ideen von AR-Anwendungen liegen.
Damit ein gewisser zeitlicher Rahmen bei der Umsetzung nicht überschritten wird, wurden Lösungen, die eine große Einarbeitungszeit oder fundierte Kenntnisse einer oder mehrer Programmiersprachen erfordern, nicht genauer betrachtet. Unter einer einfachen Umsetzung ist nicht zuletzt auch zu verstehen, dass diese auch dem Kreis der nicht-professionellen Anwender möglich sein soll. Dies beinhaltet, dass das gesuchte Programm auf normalen dem durchschnittlichen derzeitigen Stand der Technik entsprechenden Computersystemen lauffähig sein sollte. Auch wurden kommerzielle Produkte außer Acht gelassen, da die oft nicht unerheblichen Kosten einer nicht-professionellen Nutzung im Wege stehen.

Im Folgenden möchte ich mit einer kleinen Übersicht der in Frage kommenden Programme beginnen. Danach folgt dann eine Auswahl desjenigen Kandidaten, der mir für die schnelle Umsetzung eines Prototypens am vorteilhaftesten erschien. Nach einer genaueren Beschreibung von Aufbau und Funktionalität soll abschließend dessen Praxistauglichkeit anhand der Umsetzung eines vorgegebenen Szenarios geprüft werden. Abschließend erfolgt eine kurze Beurteilung, ob die gestellten Aufgaben mit Hilfe des gewählten Programms zufrieden stellend gelöst werden konnten

2 Kandidaten

Zuerst folgt eine kurze Beschreibung der jeweiligen Programme, welche in die engere Auswahl für die Umsetzung eines Prototypen kamen.

2.1 AMIRE

AMIRE (**A**uthoring **Mi**xed **Re**ality) war ein EU-finanziertes Projekt, an welchem 9 europäische Partner teilnahmen. Zielsetzung war die Entwicklung eines AR-Autorensystems, welches auch ungeübten Benutzern ohne Kenntnisse in Programmiersprachen das Erstellen und Editieren von AR-Anwendungen ermöglichen sollte. Über eine Laufzeit von 27 Monaten (01.04.2002-30.06.2004) wurde ein umfassendes Autorensystem entwickelt. (vgl.: http://www.amire.net)
Da ich mich letztendlich für die Umsetzung des Testszenarios mit Hilfe von AMIRE entschieden habe, erfolgt eine genauere Beschreibung des Funktionsumfangs in einem späteren Teil dieser Ausarbeitung.

2.2 DART

DART (**D**esigners **A**ugmented **R**eality **T**oolkit; Augmented Environments Laboratory, Georgia Institute of Technology) ist eine Sammlung von Erweiterungen für *Macromedia Director*. Eigentlich für die Entwicklung von Multimedia-Anwendungen konzipiert, ist es somit mit Director möglich, ohne genauer Kenntnisse von AR-typischen Funktionen wie *Tracking* (das Verfolgen von unter anderem auch bewegten Objekten) oder 3-D-Darstellung, relativ schnell AR-Anwendungskonzepte umzusetzen. Sollte der Funktionsumfang der DART-Sammlung nicht ausreichen, besteht zusätzlich noch die Möglichkeit, Scripte in den Sprachen „Lingo“ oder „Java Script“ selber zu erstellen.
(vgl.: http://www.gvu.gatech.edu/dart//)

2.3 DWARF

DWARF (**D**istributed **W**earable **A**ugmented **R**ealitiy **F**ramework; Technische Universität München) ist ein *Framework* (Programmgerüst, welches eine Anwendungsarchitektur vorgibt), bei dem die Datenflussarchitektur zwischen einzelnen Bausteinen im Vordergrund steht. Getrennte voneinander unabhängige Komponenten („Services“) kommunizieren anhand sogenannter *Needs* und *Abilities* zur Laufzeit miteinander. Hier wurden aber weniger konkrete Anwendungen, als vielmehr die Softwarearchitektur für eine dezentrale Entwicklung von AR-Anwendungen beachtet. (vgl.: http://campar.in.tum.de/Chair/DwarfWhitePaper)

2.4 Studierstube

Studierstube ist ein Framework, welches die Entwicklung mobiler und kollaborativer AR-Anwendungen unterstützt. Das Projekt begann schon 1996 an der Universität Wien, seit 2004 wird es von der Universität Graz weiter entwickelt. Wichtige Bestandteile von Studierstube sind zum einen *Open Tracker*, einem Trackin-Framework entwickelt an der Universität Wien, zum anderen *Open Inventor,* einem bekannten Open-Source-Szenegraphen. Dieser ist zuständig für die Datenverwaltung, Interaktion und Präsentation in Studierstube.
(vgl.: Jegust, 2007 und http://studierstube.icg.tu-graz.ac.at/)

3 Auswahl

Wie schon einführend erwähnt waren verschiedene Kriterien für die Auswahl des Testkandidaten maßgebend:

- Voraussetzung möglichst geringer Kenntnisse des Benutzers
- Kurze Einarbeitungsphase
- Ausreichende Dokumentation
- Fehlertoleranz
- Ausreichende Funktionsvielfalt
- Gute Übersichtlichkeit
- Einfache Installation
- Geringe Kosten

Ausdrücklich lag der Schwerpunkt bei der Auswahl auf *Rapid-Prototyping* (schnelle Prototypenentwicklung), wobei zu dem Zeitaufwand nicht nur das Erstellen des Prototyps, sondern eben auch das Erlernen der Handhabung des Programms und alles was damit zusammen hängt, zählt. Unter diesem Gesichtspunkt kamen Programme wie z.B. *ARToolkit*, das wohl bekannteste und am weitesten verbreitete *Toolkit* (eine Sammlung von z.B. Bibliotheken und Klassen für die Erstellung von Programmen) für AR-Anwendungen, nicht in Frage. Alleine die Einarbeitungsphase würde sich als viel zu lange erweisen, ferner sind Programmierkenntnisse unerlässlich.

Unter diesem Aspekt schied auch Studierstube aus. Zur Erstellung einer Szene müssen mindestens grundlegende Kenntnisse von Open Inventor vorausgesetzt werden. Auch steht das Fehlen einer *GUI* (Graphical User Interface) der Benutzung durch ungeübte Anwender entgegen.

DWARF wiederum konnte ebenfalls in fast keinem der oben genannten Punkte überzeugen. Allein schon der Aufwand für Installation und Einarbeitung übersteigt den hier geforderten geringen Aufwand und eine kurze Einarbeitungsphase ist nicht zu erwarten.

Anders ist da schon DART zu bewerten. Die Installation ist denkbar einfach, die Dokumentation umfassend und klar verständlich. Zahlreiche Tutorials machen einem den Einstieg einfach. Die Drehbuch- und Besetzungs-Metapher von Director verschafft auch ungeübten Anwendern einen guten Überblick.
Als Beispiel sei hier das Besetzungs-Fenster genannt, in welchem sich unter anderen z.B. ein *DART-Event*s-Reiter befindet. Unter diesem finden sich wiederum Bausteine für das Handling von Keyboard, Trackern oder auch Buttons. Dies ist zum einen übersichtlich, zum anderen sehr intuitiv und gut angeordnet.

Zwei nachteilige Aspekte waren aber letztendlich entscheidend, warum ich mich nicht für DART entschied. Einerseits erfordert der Einsatz von DART zwingend die Installation und Kenntnis des kostenintensiven Programms Macromadia Director, andererseits sind zumindest Grundkenntnisse in der Scriptsprache *Lingo*, ggf. auch *JavaScript*, unumgänglich. Zwar ist es möglich, eine kostenlose 30-Tage-Version von Director zu erhalten, dies reicht aber leider nur zur umfassenden Einarbeitung in das Programm aus. Für eine folgende Anwendung in Verbindung mit DART ist der Kauf dieses Programms erforderlich.

Wie schon eingangs erwähnt fiel die Wahl abschließend auf AMIRE. Auch hier ist die Installation schnell und einfach, die Dokumentation umfassend. Problemlos sind direkt verschiedene Anleitungen mit ersten Schritten durchzuführen. Die Oberfläche ist auch hier sehr übersichtlich und intuitiv. Bei genauerer Betrachtung erschließt sich eine erstaunliche Vielfalt an Möglichkeiten. Des Weiteren sind keinerlei zusätzliche Programme erforderlich.
Im Folgenden möchte ich jetzt die Funktionsmöglichkeiten und Bedienung von AMIRE ein wenig genauer beschreiben.

4 AMIRE

Im Folgenden möchte ich ein wenig genauer auf den Aufbau und die Funktionen des AMIRE-Frameworks eingehen. Alle Angaben beziehen sich auf AMIRE der Version 1.1, welche unter http://sourceforge.net/projects/amire/ für den Download bereit steht. Eine Installation ist nicht erforderlich, es muss lediglich eine Zip-Datei entpackt werden. Durch den Start über eine Batch-Datei ist auch kein weiteres kopieren von Systemdateien erforderlich.

4.1 Aufbau

Nach dem Start der Datei *amirev1.bat* (*...\amire_v1.1\bin\amirev1.bat*) erscheint das AMIRE-Loader-Dialogfenster (Abb.1) mit 4 Optionen:

(1) Optionale Parameter, falls eine Videodatei verwendet wird. Genaue Parametersyntax beim Klick auf das nebenstehende Fragezeichen zu ersehen.
(2) Für den Fall, dass statt Live-Bildern eine Videosequenz benutzt werden soll, kann diese hier gewählt werden. Dies ermöglicht den Betrieb von AMIRE ohne PC-Kamera.
(3) Hier können verschiedene Auflösungs-Grundeinstellungen gewählt werden.

Ferner können per Checkbox noch die Standardmarker *Hiro, Kanji* und *Sample1* bei Programmstart geladen werden.

Abbildung 1: AMIRE-Loader-Dialogfenster

Nachdem alle erforderlichen Einstellungen vorgenommen und mit dem OK-Button bestätigt wurden, erscheint ggf. noch ein kleines Fenster mit unterschiedlichen Kameratreibern. Dies ist aber nur der Fall, wenn mehr als ein Treiber bzw. mehr als eine PC-Kamera installiert wurden. Dann ist hier der entsprechende Treiber zu wählen.

Darauf folgend müssen jetzt Einstellungen in dem Videoformat-Dialogfenster (Abb.2) vorgenommen werden. Je nach verwendeter PC-Kamera sind hier die Auswahlmöglichkeiten unterschiedlich. Im Normalfall reicht es aus, eine den Anforderungen entsprechend hohe Ausgabegröße zu wählen.
Ist später ein deutliches Ruckeln der Bilder und eine sehr lange Reaktionszeit des Videobildes zu beobachten, muss die Einzelbildrate überprüft und gegebenenfalls erhöht werden.

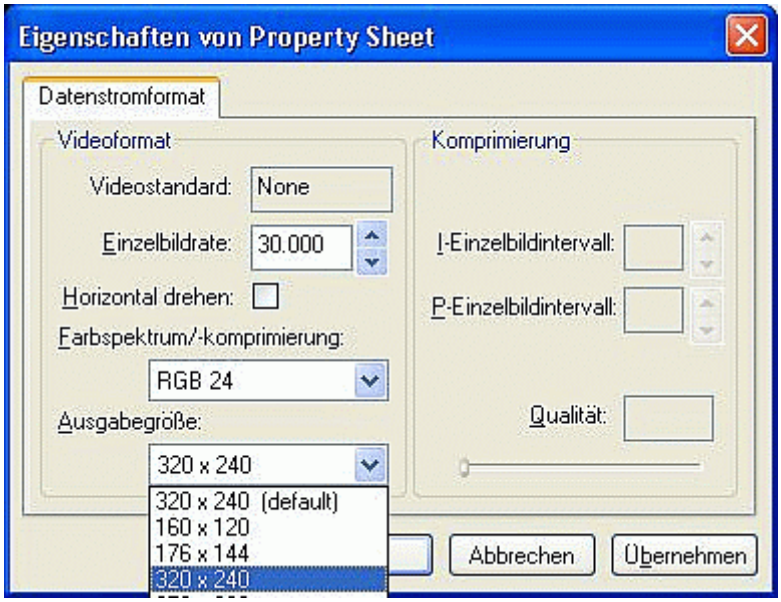

Abbildung 2: Videoformat-Dialogfenster

Nach nochmaliger Bestätigung der Einstellungen gelangen wir nun zum Hauptfenster des AMIRE-Frameworks. Dies ist wiederum aus einzelnen Fenstern aufgebaut:

(1) Das Kamerabild. Im oberen Teil befindet sich eine Menüleiste und Buttons für die wichtigsten Funktionen.
(2) Das Szenenfenster zeigt den Aufbau der gesamten Szene in einer Baumstruktur. Hier bekommt man einen schnellen Überblick über alle an der Szene beteiligten Komponenten, verwendeten Objekte, Marker etc.
(3) Das Einstellungsfenster. Hier kann für jeweilige im Szenenfenster gewählte Objekte oder Funktionen einzelne Parameter gewählt und editiert werden.
(4) Das Objektfenster. Hier sind alle zur Verfügung stehenden Objekte, Funktionen und Komponenten aufgeführt.
(5) Das Skriptfenster. Hier können über die direkte Eingabe z.B. Skripte geladen und ausgeführt werden.
(6) Das Nachrichtenfenster. Unter anderem erfolgt hier auch die Ausgabe von Fehlermeldungen.

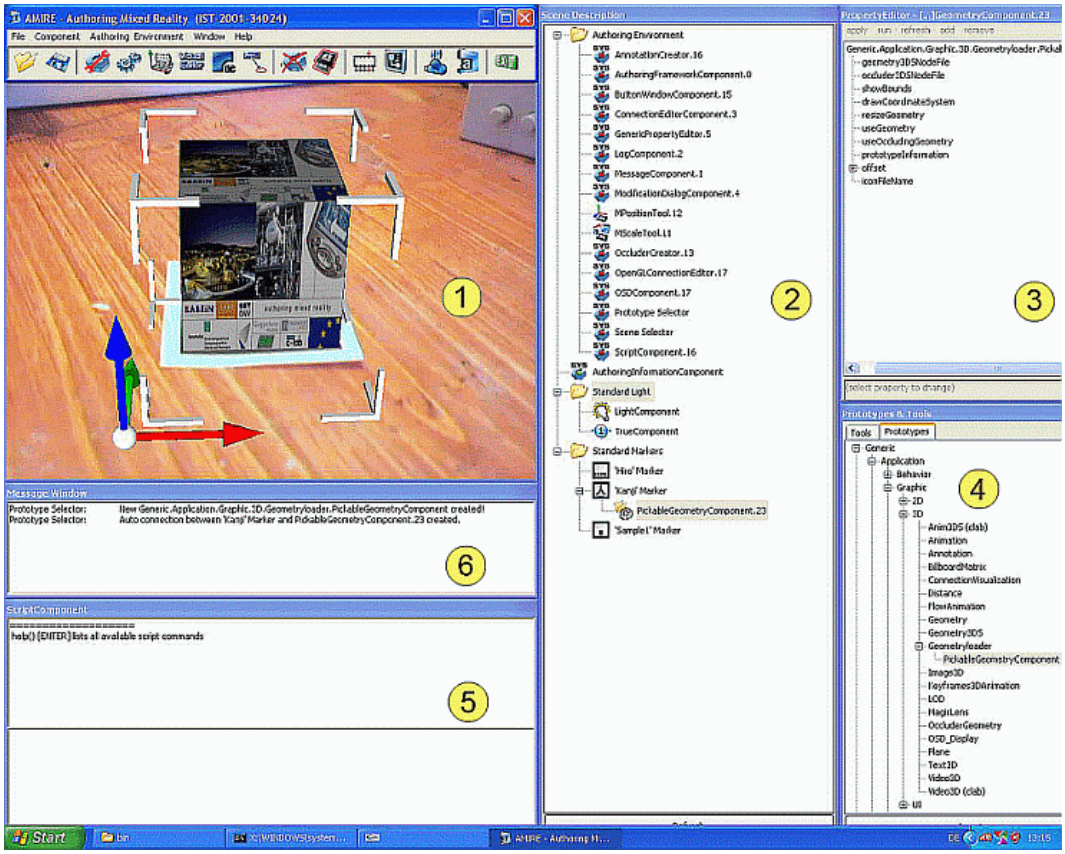

Abbildung 3: Das AMIRE-Framework

4.2 Elemente

In diesem Teil möchte ich ein wenig genauer auf den grundsätzlichen Aufbau und die wichtigsten Elemente des Frameworks eingehen. Der Funktionsumfang des AMIRE-Frameworks ist so groß, dass sicherlich an dieser Stelle nicht alles beschrieben werden kann. Daher beschränke ich mich in der Hauptsache auf Elemente, welche später für die Erstellung des Testszenarios benötigt wurden. Alle erstellten Szenen können im XML-Format abgespeichert werden.
Weitere Informationen und Tutorials sind sowohl in der Installationsdatei enthalten, als auch z.B. auf http://www.amire.net/publications_deliverables.html oder http://webster.fh-hagenberg.at/amire/development.html erhältlich.
Die grundlegenden Elemente, aus denen eine Szene aufgebaut werden kann, habe ich in Objekte, Funktionen und Komponenten unterteilt. Die Auswahl aller Elemente erfolgt über das Objektfenster. Eingefügt wird ein Element entweder über den Insert-Button des Objektfensters, oder über einen Doppelklick auf das ge-

wählte Element. Grundsätzlich wird ein Element an der markierten Stelle im Szenenfenster angefügt. Sollte dies nicht vorgesehen sein, wird automatisch ein anderer Ort im Baum gewählt.
Jedes Element verfügt über sogenannte *Inslots* und *Outslots*, über diese werden Elemente auch miteinander verbunden. Ein einfaches Beispiel hierfür ist das Verbinden eines Markers mit einem Objekt wie z.B. *Image3D* oder auch einer *PickableGeometryComponent*. Nach markieren des gewählten Markers wird ein Image3D (Generic/Application/Graphic/3D/Image3D) per Doppelklick angefügt. Automatisch werden die Outslots *visible* und *transformation* des Markers mit den Inslots *visible* und *transformation* des 3-D-Bildes verbunden. Das 3-D-Bild wird somit sichtbar, wenn der Marker sichtbar ist. Auf den Marker angewandte Transformationen werden auch auf das Bild angewandt. Soll die Verbindung der Slots nicht automatisch erfolgen, muss der *AutoConnect-Enabled*-Button des Objektfensters betätigt werden.
Jederzeit können bestehende Verbindungen über den *Connection-Edit*or überprüft und editiert werden. Hierfür markiert man zwei Elemente im Szenenfenster und ruft den Connection-Editor über die Button-Leiste oberhalb des Kamerabildes auf. In zwei Fenstern sind nun die Inslots des einen und die Outslots des anderen Elementes zu sehen. Werden zwei Slots markiert, können diese über die *Connect/Disconnect*-Buttons in der Mitte verbunden oder getrennt werden.

4.2.1 Objekte

Image3D (Generic/Application/Graphic/3D/Image3D) fügt ein 3-D-Bild in die Szene ein. Über das Einstellungsfenster können *imageFileName*, *size*, *offset* und *iconFilename* editiert werden.

- imageFilename: per Doppelklick können in einem Dialogfenster alle gängigen Bilddateiformate geladen werden.
- size: die Bildgröße kann in der Einstellungsfenster-Kommandozeile geändert werden
- offset: per Doppelklick öffnet sich der *Matrix-Editor*. Hier kann das Bild verschoben, skaliert und rotiert werden. Dies erfolgt entweder über Schieberegler für die x-, y-, z-Werte, oder über direkte Eingabe eines Wertes. Ferner kann auch Gesamtmatrix (*Offset Matrix*) durch Eingabe von Werten verändert werden.
- iconFileName: hier kann eine Bilddatei für das Erscheinungsbild im Szenenbaum gewählt werden. Ist keine Datei angegeben, wird ein blauer Kreis mit Fragezeichen eingefügt.

Geometry3DS (Generic/Application/Graphic/3D/Geometry3DS) fügt ein 3-D-Objekt in die Szene ein. Diese Objekte können bei Bedarf in dem Modellierungs- und Animationsprogramm *3-D Studio Max* (Autodesk) erstellt werden. Über das Einstellungsfenster können imageFilename, size, offset und iconFilename editiert werden. dies kann auf gleiche Weise wie für Image3D vorgenommen werden.

Button3D (Generic/Application/Graphic/UI/Button3D) fügt einen 3-D-Button in die Szene ein. Über das Einstellungsfenster können *releasedImageFilename*, *toggledImageFilename*, *size*, *offset* und *iconFileName* editiert werden.

- releasedImageFilename: per Doppelklick kann eine Bilddatei für den nicht betätigten Button gewählt werden.
- toggledImageFilename: per Doppelklick kann eine Bilddatei für den betätigten Button gewählt werden.
- size: die Größe des Buttons kann über die Kommandozeile des Einstellungsfensters gewählt werden.
- offset: per Doppelklick öffnet sich der *Matrix-Editor*. Hier kann das Bild translatiert, skaliert und rotiert werden. Dies erfolgt entweder über Schieberegler für die x-, y-, z-Werte, oder über direkte Eingabe eines Wertes. Ferner kann auch Gesamtmatrix durch Eingabe von Werten verändert werden.

Button2D (Generic/Application/Graphic/UI/Button2D) fügt einen 2-D-Button in die Szene ein. Dieser kann aber nicht mit einem Marker verbunden, sondern nur absolut in das Kamerabild-Fenster eingefügt werden. Daher erscheint er auch direkt im Szenenbaum und wird nicht mit einem anderen Objekt verbunden. Im Einstellungsfenster kann auch nicht der Matrix-Editor aufgerufen werden, stattdessen besteht die Möglichkeit, Werte für *position*, *dimension* und *alignment* zu verändern.

- position verändert die Position des Buttons über x- und y-Wert durch Eingabe über die Einstellungsfenster-Kommandozeile.
- dimension verändert die Größe des Buttons über x- und y-Wert durch Eingabe über die Einstellungsfenster-Kommandozeile.
- alignment verändert die Ausrichtung des Buttons über x- und y-Wert durch Eingabe über die Einstellungsfenster-Kommandozeile

Text3D (Generic/Application/Graphic/3D/Text3D) fügt einen Text in die Szene ein. Die wichtigsten Einstellungen sind hier *textFileName*, *showBackground*, *panelWidth*, *panelHeight* und *offset*.

- textFilename: über Doppelklick kann hier eine Textdatei gewählt werden.
- showBackground: ist dieser Wert in der Kommandozeile auf *true* gesetzt, wird der Text auf einem halbtransparenten Hintergrund abgebildet, auf *false* gesetzt erscheint nur der reine Text.
- panelWidth: in der Kommandozeile kann hier die Breite der Fläche, auf der der Text abgebildet wird, eingestellt werden.
- panelHeight: in der Kommandozeile kann hier die Höhe der Fläche, auf der der Text abgebildet wird, eingestellt werden.

- offset: per Doppelklick kann über den Matrix-Editor die Textfläche Verschoben, rotiert und skaliert werden.

Es besteht neben den oben genannten Objekten noch die Möglichkeit, eine Vielzahl weiterer Objekte einzufügen. Neben einer *Plane*, *MagicLens* und *Annotation*, können auch komplette Videos oder Animationen in der Szene abgebildet werden. Eine Aufzählung aller Möglichkeiten mit genauer Funktionsbeschreibung würde den Rahmen dieser Arbeit sprengen.

4.2.2 Funktionen

Logic (Generic/Application/Math/Logic) kann verschiedene Elemente über eine logische Verknüpfung miteinander verbinden. Es stehen *And*, *Or*, *Not*, *Delay*, *Remain* und *Logical* zur Verfügung. Ein einfaches Beispiel für die Anwendung einer And-Funktion wäre z.B. das Verbinden der Visible-Outslots zweier Marker mit jeweils einem Inslot der And-Funktion. Deren Outslot wiederum könnte mit einem beliebigen Objekt verbunden werden, welches erst bei Sichtbarkeit beider Marker visualisiert werden würde.

- And besitzt zwei Inslots, einen Outslot und entspricht einer Und-Funktion.
- Or besitzt zwei Inslots, einen Outslot und entspricht einer Oder-Funktion.
- Not besitzt einen Inslot, einen Outslot und entspricht einer Not-Funktion.
- Delay besitzt einen Inslot, einen Outslot und führt eine Verzögerung bei der Weitergabe eines Wertes von In- zu Outslot herbei. Diese kann im Einstellungsfenster für den Wert *delay* in Frames angegeben werden.
- Remain besitzt einen Inslot., einen Outslot und führt ein Beibehalten eines Wertes am Outslot herbei, wenn sich dieser am Inslot geändert hat. Die Länge kann im Einstellungsfenster für den Wert *remain* in *Frames* (Bilder pro Sekunde) angegeben werden.
- Logical besitzt eine variable Anzahl an Inslots, die Anzahl ist über den Wert *numElements* in der Kommandozeile anzugeben. Ferner ist die Art der Logik variabel, es kann über den Wert *option* zwischen AND, OR, NAND, NOR, XOR und NOT gewählt werden.

Distance (Generic/Application/Graphic/3D/Distance) errechnet die Distanz zwischen zwei Objekten. Bei Bedarf können die Objekte mit einer Linie verbunden und die Distanz in Millimeter im Kamerabild eingeblendet werden, ferner kann man sich noch die Distanz in Millimeter und den Winkel der Objekte zueinander in Grad in einem Fenster anzeigen lassen. Wird nur ein Objekt gewählt, ist die gemessene Distanz die zwischen Objekt und Kamera. Wichtige Einstellungen sind hier *vline*, *vchar*, *markerSize*, *useCamera* und *useOSD*.

- vline blendet je nach Wert true oder false in der Kommandozeile eine Linie zwischen den Objekten ein.

- vchar blendet je nach Wert true oder false in der Kommandozeile die Distanz der Objekte im Kamerabild ein.
- markerSize ist für die Eingabe der korrekten Markergröße, eine falsche Eingabe hat eine falsche Schätzung der Distanz zur Folge.
- useCamera muß auf true gesetzt werden, falls nur ein Objekt mit dem Distance-Element verbunden ist und die Entfernung zur Kamera angezeigt werden soll.
- useOSD blendet je nach Wert true oder false in der Kommandozeile ein On-Screen-Display ein.

BillboardMatrix (Generic/Application/Graphic/3D/BillboardMatrix) errechnet aus der Transformationsmatrix eines Objektes eine Billboard-Matrix. Dies hat zur Folge, dass das Objekt stets zur Kameraebene ausgerichtet ist. Die einzige Einstellmöglichkeit ist *offset.* Hier kann über den Matrix-Editor die Ausrichtung des Objektes verändert werden. Die Billboard-Matrix wird zwischen zwei Transformation-In- und Outslots geschaltet.

LOD (Generic/Application/Graphic/3D/LOD) ist eine Level of Detail- Funktion, welche abhängig von einer Distanz drei verschiedene Objekte visualisieren kann. LOD kann nur in Verbindung mit der Distanz-Funktion benutzt werden, da einer der Inslots die Distanzinformation benötigt. Im Einstellungsfenster können für die Werte *minimum* und *maximum* Entfernungen eingegeben werden, die einen Bereich eingrenzen. Haben wir als Beispiel über die Distanz-Funktion einen Marker und die Kamera verbunden, und die Entfernung der Kamera zum Marker liegt in dem mit minimum und maximum eingegrenzten Bereich, wird der LOD-Outslot *show* auf true gesetzt. Wird dieser Bereich unterschritten, wird *showless* auf true gesetzt, wird er überschritten, ist *showmore* auf true gesetzt. Haben wir nun diese drei Outslots mit den Visible-Inslots verschiedener Objekte verbunden, wird je nach Entfernung der Kamera zum Marker ein anderes Objekt visualisiert.

4.2.3 Komponenten

PickableGeometryComponent (Generic/Application/Graphic/3D/ GeometryLoader/PickableGeometryComponent) bietet die Möglichkeit, 3DS-Dateien (3-D Studio Max) in die Szene einzubinden. Ferner sind diese Objekte mit der Maus im Kamerabild wählbar. Im Einstellungsfenster kann man folgende Werte editieren:

- *geometry3DSNodeFile*: per Doppelklick kann hier eine 3DS-Datei gewählt werden, die auf übliche Art und Weise abgebildet wird.
- *occluder3DSNodeFile*: hier kann eine 3DS-Datei gewählt werden, welche für eine Überdeckung verwendet wird.

- *showBounds*: bildet je nach true/false-Wert die *Bounding Sphere* (eingrenzendes Volumen) der Geometrie ab.
- *drawCoordinateSystem* zeigt je nach true/false-Wert das Koordinatensystem der Geometrie an.
- *resizeGeometry* skaliert die Geometrie auf einen Maximalwert von 80x80x80
- *useGeometry* verwendet je nach true/false-Wert die unter geometry3DSNodeFile angegebene Datei.
- *useOccludingGeometry* verwendet je nach true/false-Wert die unter occluder3DSNodeFile angegebene Datei
- *offset* ruft per Doppelklick den Matrixeditor für die Geometrie auf.

CentralStateComponent (Generic/Application/Behavior/
CentralStateComponent) ist eine Komponente, mit Hilfe derer man komplexere AR-Szenen in unterschiedliche Zustände aufteilen kann. Es werden Zustände und Übergänge definiert, ein Zustand (*State*) besitzt Auslöser (*Activators*) und Abhängige (*Dependants*). Im Folgenden möchte ich kurz ein wenig genauer auf diese Komponente eingehen, da ihr Einsatz für größere AR-Anwendungen unerlässlich ist.
Einmal eingefügt in den Szenenbaum lässt sich die CentralStateComponent jederzeit über die Leiste oberhalb des Kamerabildes wieder öffnen. Durch markieren eines noch nicht benannten States (1) kann dieser benannt werden (2). Erst durch Bestätigung (3) wird dies aber auch tatsächlich übernommen. Nachdem alle erforderlichen States angelegt wurden, können jetzt die gewünschten auslösenden Objekte (Auslöser) (4) und in den entsprechenden States abgebildete Objekte (Abhängige) (5) editiert werden.
Grundsätzlich ist zu beachten, dass Objekte bestimmten States erst zugewiesen werden können, wenn sie vorher schon auf die übliche Art und Weise in die Szene eingefügt wurden. Die Verbindung einzelner Objekte mit der CentralStateComponent erfolgt über zwei getrennte Auswahlfenster, Auslöser bewerkstelligen dies durch die Verbindung ihrer Outslots, Abhängige durch Verbindung ihrer Inslots mit der CentralStateComponent.

Durch Betätigen des Activator-Buttons (4) öffnet sich das Activator-Auswahlfenster für den entsprechenden State. Die Unterteilung der Auslöser ist hier in *Reality-Dependant-Activator*, *Virtuality-Dependant-Activator* und *Internal-Dependant-Activator* vorgenommen. Alle können aber gleichermaßen markiert und über den Auswahl-Button dem *Current-State-Activators-Fenster* zugefügt werden. Es ist zu beachten, dass die Angabe mehrerer Auslöser zur Folge hat, dass jeder Auslöser für sich alleine den entsprechenden State aktivieren kann. Ist gewünscht, dass z.B. ein State erst bei Sichtbarkeit von drei Markern gleichzeitig ausgelöst wird, sind diese vorher entsprechend zu verbinden (Logic), um danach diese Verknüpfung als Auslöser anzugeben.

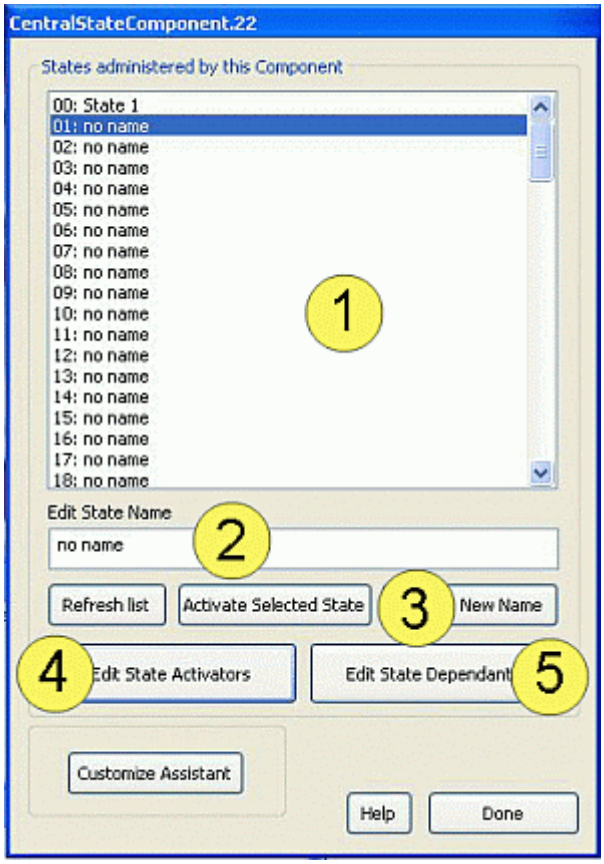

Abbildung 4: CentralStateComponent-Auswahl

Durch betätigen des Dependants-Button öffnet sich das Dependants-Auswahlfenster für den entsprechenden State. Hier kann nur in einem Bereich zwischen allen in der Szene vorhandenen Kandidaten ausgewählt werden.
Grundsätzlich ist zu beachten, dass Objekte direkt oder indirekt eingefügt werden können. Wird ein Objekt über den Button „Add selected DIRECTLY“ ausgewählt, so wird er unabhängig von anderen Bedingungen im entsprechenden State in der Szene visualisiert. Erfolgt die Auswahl aber über den Button „Add selected INDIRECTLY“, ist die Visualisierung zusätzlich noch von einem weiteren existierenden visible-Inslot abhängig. Eine AND-Verknüpfung wird automatisch generiert. Im Normalfall wird diese Option gewählt, wenn ein Objekt in einem bestimmten State sichtbar werden soll, dies aber gleichzeitig von der Sichtbarkeit des für die Transformation dieses Objektes verantwortlichen Markers abhängig gemacht werden muss.

AudioComponent (Generic/Application/Misc/AudioComponent) ermöglicht die Einbindung einer Audio-Datei in die Szene. Es ist sowohl ein Aktivieren (*activate*), wie auch ein Deaktivieren (*Stop*) der Komponente über die entsprechenden Inslots möglich. Es können Einstellungen für *audioFilename*, *loop* und *debug* im Einstellungsfenster vorgenommen werden.

- audioFilename: hier kann per Doppelklick eine Audiodatei ausgesucht werden. Das Abspielen ist in den Formaten WAV, MP3 und WMA möglich.
- loop bewirkt je nach true/false-Wert, dass die Audiodatei nur einmal oder aber in einer Endlosschleife abgespielt wird.
- debug ermöglicht, wenn auf true gesetzt, dass die Audiodatei unabhängig von der Aktivierung der Inslots abgespielt wird.

4.3 Marker

Wie unter 4.1 schon erwähnt, ist es möglich, direkt bei Programmstart die drei Standardmarker hiro, kanji und sample1 zu laden und in den Szenenbaum zu integrieren. Da aber nicht zuletzt bei größeren und komplexeren Anwendungen eine größere Anzahl benötig wird, gibt es noch die Option, weitere Marker über das Scriptfenster aufzurufen. Dies erfolgt durch Eingabe des Befehls „run(„../data/script/modelmarkers.txt“) [Enter]“.
Nach erfolgreicher Ausführung des Scripts stehen 24 weitere Marker zur Verfügung. Sowohl die Standardmarker, als auch durch das Script geladenen Zusatzmarker sind unter *amire_v1.1\doc\AMIRE-Pattern-Collection.pdf* zu finden und können ausgedruckt werden.
Ist es nicht erforderlich, eine große Anzahl an Markern per Script zu laden, können einzelne Marker über das Einfügen einer *PickableMarkerComponent* (Generic/Application/Tracking/Marker/Singlemarker/PickableMarkerComponent)
zugefügt werden. Wie bei jedem anderen Marker, sind auch hier folgende Einstellmöglichkeiten zu finden:

- *markerWidth*: hier sollte eine möglichst genaue Größenangabe gemacht werden. Eine falsche Größe hat zwangsläufig eine falsche Entfernungsschätzung zur Folge.
- *markerFile*: hier kann per Doppelklick eine entsprechende Pattern-Datei gewählt werden.
- *detectedHintFile*: hier kann per Doppelklick eine Bilddatei gewählt werden, welche auf dem detektierten Marker eingeblendet wird. Ist hier keine Datei angegeben, wird ein Hinweis in Form einer roten OpenGL-Fläche eingeblendet.

- *showPickingBound* zeigt je nach true/false-Wert anhand einer eingeblendeten Umrandung, in welchem Bereich der Marker mit der Maus ausgewählt werden kann.
- *showDetectedHint* blendet je nach true/false-Wert eine rote OpenGL-Fläche oder eine unter detectedHintFile angegebene Bilddatei ein, wenn der Marker detektiert wurde.
- *pickable* entscheidet über einen true/false-Wert, ob ein Marker mit der Maus ausgewählt werden kann.
- *dampFac* glättet je nach Wert die Bewegung der auf den Marker eingeblendeten Objekte, wenn die Position des Markers verändert wird.
- *offset* ruft den Matrixeditor auf, mit dem die Position der Picking-Bound, und des DetectedHint verändert werden kann.

Neben der Benutzung der vorgegebenen Marker bietet AMIRE zusätzlich die Möglichkeit, eigene Marker mit Hilfe des Marker-Werkzeuges zu erstellen. Dazu ist die *mk_patt.exe* (amire_v1.1\tools\mk_patt.exe) auszuführen. Per Kommandozeile muss nun der Pfad zu der Datei *camera_para.dat* (amire_v1.1\data\cameras\camera_para.dat) angegeben werden, wahlweise ist mk_patt.exe im gleichen Ordner auszuführen, dann reicht der Name der Parameterdatei aus. Danach ist der neu erstellte Marker mit dem Kamerabild zu erfassen. Geschieht dies erfolgreich, was an einer dünnen roten Umrandung des Markers zu erkennen ist, muss dieser mit der Maus gewählt werden. Als letztes ist noch ein Name anzugeben. Die neu erstellte Marker-Datei ist im Ordner der mk_patt.exe zu finden und kann direkt als Datei unter markerFile angegeben werden.

5 Testszenario

Nach der theoretischen Auseinandersetzung mit den verschiedenen Kandidaten und den Möglichkeiten des AMIRE-Frameworks, soll nun eine praktische Umsetzung folgen. Erstellt wurde hierfür ein Testszenario, welches eine virtuelle Mensamenü-Karte für eine Woche repräsentieren soll. Folgende Bestandteile sollte das Szenario beinhalten:

- Menüübersicht der Speisen einer Woche mit Bildern
- Auswahlmöglichkeit zwischen vegetarischen, nicht-vegetarischen Menüs
- Auswahlmöglichkeit verschiedener Getränke
- Übersicht über gewählte Speisen und Getränke und Preisinformation
- Regelbarer Bräunungsgrad der Pommes-Frites per Schieberegler

Denkbarer Platz für diese virtuelle Mensakarte wäre z.B. die Außenseite des Mensagebäudes oder als Erweiterung neben der üblichen Papiermensakarte.
Ob diese Umsetzung nachher z.B. per PC-Kamera und Laptop/*PDA* (Personal Digital Assistant) oder per *HMD* (Head Mounted Display) erfolgt, ist zunächst einmal für die grundsätzliche Erstellung des Prototyps und der Prüfung seiner Funktionalität unerheblich.
Maßgebend für diese Arbeit ist in erster Linie, ob und mit welchem Aufwand eine Realisierung möglich ist und in wie weit dies auch für Personen mit eingeschränkten Programmierkenntnisse ohne umfassenderen „AR-Hintergrund“ umzusetzen ist.

6 Umsetzung mit AMIRE

Dieser Ausarbeitung beiliegend ist eine CD-ROM, auf der sowohl ein Ordner (*additional*) mit dem entwickelten Prototyp, wie auch eine PDF-Datei mit der dafür erforderlichen Markeranordnung zu finden ist. Die Datei *menue2.1.xml* (additional/XML/menue2.1.xml) ist aus dem Kontextmenü von AMIRE heraus zu öffnen, die erforderlichen Bild- und Sounddateien (additional/Menue1) werden selbständig geladen.

6.1 Interaktion

Bevor genauer über eine Menüstruktur nachgedacht werden kann, muss erst einmal festgelegt werden, auf welche Art und Weise der Benutzer mit dem Menü interagieren kann und mit welchen Methoden dies vom System registriert wird. Die verschiedenen Möglichkeiten, die sich in diesem Bereich bieten, können grundsätzlich in zwei Kategorien unterteilt werden: *In-side-out* , wobei sich hier die Kamera(s) am Körper befindet, und *Out-side-in*, wogegen sich dabei die Kamera(s) in der Umgebung befinden (vergl. Müller, 2004:). Da in diesem Stadium des Prototyps noch keine genauere Festlegung erfolgte, über welche Visualisierungsmöglichkeiten der Benutzer verfügt (HMD, Laptop, PDA etc.), bietet sich hier eindeutig eine In-side-out-Lösung an. Zudem ist auch die genauere Position des Menüs in der Praxis später noch nicht festgelegt. Das könnte unter Umständen einen erheblichen Aufwand bei der Anbringung externer Kameras nach sich ziehen.
Auch wenn später die Umsetzung komplexerer Trackingverfahren durchgeführt werden kann, ist doch erst einmal zu überlegen, welche Möglichkeiten sich mit dem AMIRE-Framework bieten. Da hier ohne größeren Aufwand nur ein Tracking konventioneller Marker möglich ist, bieten sich dadurch nur noch die Umsetzung der Interaktion über Sichtbarkeit einzelner Marker. Folgend möchte ich kurz die für mich in Frage gekommenen Konzepte nennen:

- **Interaktion mit einem Zeigestift.** Hierbei wäre denkbar gewesen, einen herkömmlichen Kugelschreiber oder ähnliches mit einem Marker auszustatten. Über die in AMIRE integrierte Distanz-Funktion hätte das drücken eines Knopfes über die Entfernung das Stiftes zu dem entsprechenden Marker erfolgen können. Hiefür hätte der Stift-Marker aber schon an der Spitze des Stiftes angebracht werden müssen, da sonst je nach Position mehrere Knöpfe gleichzeitig hätten ausgelöst werden können.
- **Interaktion mit der Hand**. Dafür wäre es wohl am einfachsten gewesen, einen Handschuh mit aufgeklebten Markern zu benutzen. Grundsätzlich wäre die Interaktion nicht anders als bei dem Zeigestift. Dafür wären die Möglichkeiten durch Anbringung verschiedener Marker an den Fingern aber noch erweiterbar gewesen. Idee: Die Auswahl erfolgt durch Annäherung mit dem Zeigefinger, das Drücken eines Knopfes erst durch Annä-

hern des Mittelfinger-Markers an den Zeigefinger-Marker. Aber auch hier müssten die Menüknöpfe weit genug auseinander sein, ferner verlangt diese Idee deutlich mehr Verständnis des Benutzers.

- **Interaktion mit der Hand durch Verdeckung.** Bei dieser Idee erfolgt das Auslosen nicht durch Sichtbarkeit eines Markers, sondern durch Verdeckung eines Markers.

Ganz klarer Vorteil der Interaktion durch Verdeckung ist, dass keinerlei Hilfsmittel benötigt werden. Auch verspricht diese Idee ein genaueres Interagieren, da eine Verdeckung zweier Marker gleichzeitig sehr leicht durch den Benutzer zu kontrollieren und zu verhindern ist. Aus diesen Gründen entschied ich mich, die dritte Möglichkeit in das Testszenario zu integrieren.
Nachteil dieser Möglichkeit ist, dass der Marker, über den die Verdeckung registriert werden soll, sich von dem Marker, der den gedrückten Knopf visualisieren muss, unterscheiden sollte. Eine Verdeckung hätte sonst zwangsläufig das Verschwinden des gedrückten Knopfes zur Folge.

6.2 States

Als nächstes war die Frage, wie der Grundaufbau des Menüs auszusehen hatte. Bedingt durch die Interaktion mit nur einem Menü an einer Stelle war klar, dass möglichst wenige Marker für alle Interaktionen und Visualisierungen benutzt werden müssen. Bei Integration aller Bestandteile in eine Szene trat sehr schnell das folgende Problem auf: Für jede Aktion, die durch Verdeckung eines Markers ausgelöst werden sollte, musste ein anderer Marker benutzt werden. Ferner war es kein Problem, die Speisen eines Wochentages bei Verdeckung eines Markers einzublenden, aber bei zurückkehrender Sichtbarkeit des Markers wurden diese wieder ausgeblendet. Zwar konnte dies durch Zwischenschalten einer Remain-Funktion verhindert werden, spätestens bei der Einblendung einer weiteren Speise kam es aber zu Überlappungen.
Genau hierfür ist der Einsatz der CentralStateComponent nötig. Die Unterteilung der Szene in verschiedene States, wobei jeder State die Visualisierung eines anderen Wochentages übernimmt, hat unter anderem zur Folge, dass eine deutlich geringere Anzahl an Markern benutzt werden muss. Der Auslöser für einen State1 z.B. kann die Verdeckung eines Marker1 und gleichzeitig die Sichtbarkeit eines Objektes1 sein. Wird in State1 nun ein Objekt2 visualisiert, kann hier wiederum die Verdeckung des Marker1 und gleichzeitig die Sichtbarkeit eines Objektes2 der Auslöser für einen State2 sein. In diesem Fall kann also immer wieder der gleiche Marker benutzt werden und z.B. die Funktion eines „Vorwärts-Buttons“ übernehmen. In Tabelle 1 gebe ich eine Übersicht über alle angelegten States, ihre Auslöser und Abhängige.

States	Activator	Dependants
State00: Start_Mittwoch	AND_to_State_0_from_State_1 AND_to_State_0_from_State_4 AND_to_State_0_from_State_5 AND_to_State_0_from_State_10 AND_to_State_0_from_State_11 AND_to_State_0_from_State_12	Delay_Mittwoch, Button_vegetarisch Button_getraenke BillboardMatrix Delay_Audio_Menue
State01: Donnerstag	AND_to_State_1_from_State_0 AND_to_State_1_from_State_2 AND_to_State_1_from_State_6	Delay_Donnerstag Button_vegetarisch Button_getraenke BillboardMatrix
State02: Freitag	AND_to_State_2_from_State_1 AND_to_State_2_from_State_3 AND_to_State_2_from_State_7	Delay_Freitag Button_vegetarisch Button_getraenke BillboardMatrix
State03: Montag	AND_to_State_3_from_State_2 AND_to_State_3_from_State_4 AND_to_State_3_from_State_8	Delay_Montag Button_vegetarisch Button_getraenke BillboardMatrix
State04: Dienstag	AND_to_State_4_from_State_3 AND_to_State_4_from_State_9 AND_to_State_4_from_State_0	Delay_Dienstag Button_vegetarisch BillboardMatrix Button_getraenke
State05: Mittwoch_vegi	AND_to_State_5_from_State_0 AND_to_State_5_from_State_9 AND_to_State_5_from_State_6	Delay_Mittwoch_vegi Button_nicht_vegetarisch Button_getraenke BillboardMatrix
State06: Donnerstag_vegi	AND_to_State_6_from_State_5 AND_to_State_6_from_State_7 AND_to_State_6_from_State_1	Delay_Donnerstag_vegi Button_nicht_vegetarisch Button_getraenke BillboardMatrix
State07: Freitag_vegi	AND_to_State_7_from_State_6 AND_to_State_7_from_State_8 AND_to_State_7_from_State_2	Delay_Freitag_vegi Button_nicht_vegetarisch Button_getraenke BillboardMatrix
State08: Montag_vegi	AND_to_State_8_from_State_7 AND_to_State_8_from_State_9 AND_to_State_8_from_State_3	Delay_Montag_vegi Button_nicht_vegetarisch Button_getraenke BillboardMatrix
State09: Dienstag_vegi	AND_to_State_9_from_State_8 AND_to_State_9_from_State_5 AND_to_State_9_from_State_4	Delay_Dienstag_vegi Button_nicht_vegetarisch Button_getraenke BillboardMatrix

State10: Getraenke_Mittwoch	AND_to_State_10_from_State_0 AND_to_State_10_from_State_11 AND_to_State_10_from_State_11	Delay_Getraenk Button_zurueck Button_weiter BillboardMatrix
State11: Getraenke2_Mittwoch	AND_to_State_11_from_State_10 AND_to_State_11_from_State_12	Delay_Getraenk1 Button_zurueck Button_weiter BillboardMatrix
State12: Getraenke3_Mittwoch	AND_to_State_12_from_State_11 AND_to_State_12_from_State_10	Delay_Getraenk2 Button_zurueck Button_weiter BillboardMatrix
State13: Pommes_Auswahl	AND_to_State_13_from_State_11	Delay_Auswahl Button_kauf_ich LOD Distance BillboardMatrix
State14: Ende	AND_to_State_14_from_State_13	Audio_Component_Danke Text3D

Tabelle 1: State-Übersicht

6.3 Funktionalität

Als nächstes folgt nun die genauere Ausarbeitung der Menü-Elemente und deren Anordnung. Zuerst soll der Benutzer die Speisen der verschiedenen Wochentage, sowohl vegetarische als auch nicht-vegetarische, betrachten können. Um dies ein wenig ansprechender zu gestalten, wurde diese Ansicht als eine Art Ring-Menü angelegt, in dem neben einer großen Ansicht des Tagesmenüs die anderen Tage in kleineren Abbildungen sichtbar sind. Ein Wechsel zwischen den Tagen erfolgt über zwei Pfeiltasten, mit denen in beide Richtungen das Auswahlmenü weitergedreht werden kann. Ferner soll ein Wechsel zwischen den vegetarischen und nicht-vegetarischen Menüs jederzeit möglich sein.
Ist die Auswahl des Menüs abgeschlossen, kann über einen weiteren Button zur Getränkeauswahl gewechselt werden. Auch hier ist die Übersicht über die Getränke als Ring-Menü gestaltet. Die Pfeiltasten haben hier wieder die gleiche Funktion. Ist auch die Auswahl der Getränke abgeschlossen, soll in einem nächsten Schritt noch eine Übersicht über die gewählten Speisen und Getränke gezeigt werden. Zusätzlich kann noch der Bräunungsgrad der Pommes-Frites geändert werden. Ist auch dieser Schritt beendet, wird die Menüwahl mit dem Einblenden einer Kostenauflistung abgeschlossen.
Neben der Visualisierung wurde auch noch eine Audio-Komponente hinzugefügt.
Im nächsten Kapitel möchte ich ein wenig näher auf die technische Umsetzung und den damit verbundenen Problemen eingehen.

6.4 Umsetzung

Zuerst möchte ich einen kurzen Überblick über die Anordnung der Marker und ihrer Funktion geben (Abb.5). In der Mitte oben befindet sich der Hauptmarker für die Abbildung des Ring-Menüs. Unterhalb davon links und rechts von der Mitte sollen sich die Pfeiltasten befinden. Der Auswahlknopf vegetarisch/nicht-vegetarisch ist wieder in der Mitte angebracht, aber etwas oberhalb der Pfeiltasten, damit er bei einem Wechsel von links nach rechts nicht versehentlich betätigt wird. In gleicher Höhe aber auf der rechten Seite (in diesem Fall ist das Menü für Rechtshänder ausgelegt) wurde der Button platziert, mit dem im Gesamtmenü sich vorwärts bewegt. Die Größe der Marker ist bis auf den Hauptmarker klein gehalten damit sie weitgehend durch die Objekte verdeckt werden.

(1) *Pattern Sample1*: Ring-Menü für Essen (veg. u. nicht-veg), Ring-Menü für Getränke, zusammenfassende Abbildung und Rechnung.
(2) *Pattern1*: Pfeiltaste links
(3) *Pattern2*: Pfeiltaste rechts
(4) *Pattern3*: Button „Vegetarisch“ und „Nicht Vegetarisch“, Button „Zurück“ und Abbildung Pommes-Frites
(5) *Pattern4*: Button „Getränke“, Button „Weiter“ und Button „Kauf ich!“

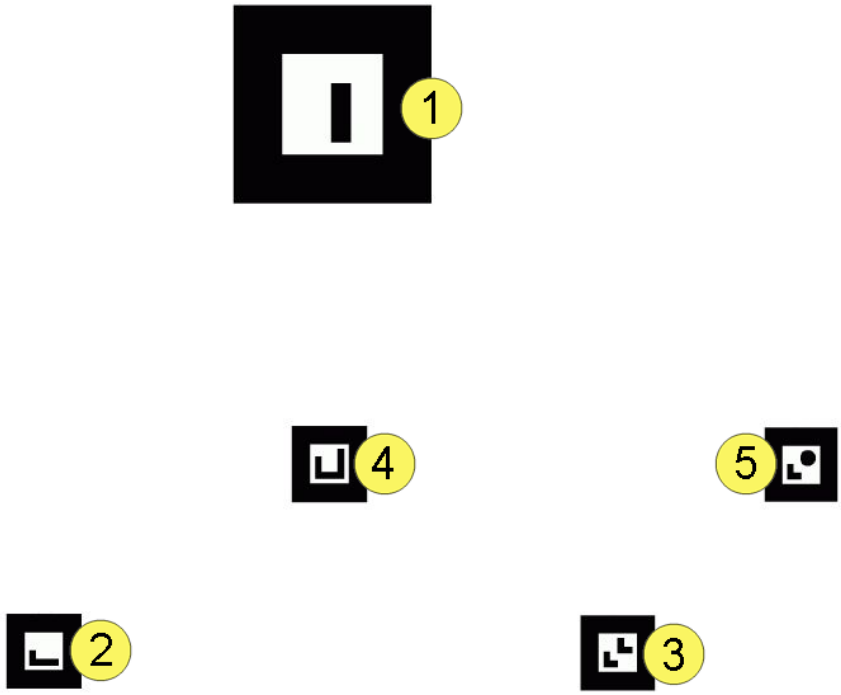

Abbildung 5: Marker-Anordnung

Beginnen wir nun mit der Anordnung der eigentlichen Elemente der Prototyps (Abb. 6), zuerst mit dem **Ring-Menü** (1). Die Speise des entsprechenden Tages ist in der Mitte und groß abgebildet. Um die Struktur einfach zu halten wird nur dieses eine Bild mit dem Marker verbunden und steht auch in den jeweiligen States für das gesammte Ring-Menü eines Tages. Die vier kleineren Abbildungen der anderen Tage werden relativ zu diesem einen Bild visualisiert, wobei Verschiebung, Rotation und Skalierung über den Matrixeditor erfolgen. Für alle Tage wurden als Bilder Platzhalter verwendet (die Farbe Weiß entspricht nicht-vegetarisch, Grün vegetarisch), einzig der Mittwoch wurde mit einer richtigen Abbildung einer Speise ausgestattet. Dies hat den Grund, dass zum einen eine Auswahl zwischen den Speisen einer ganzen Woche nicht sinnvoll wäre, zum anderen ein Gang in die Mensa der Universität Koblenz sich an einem anderen Tag als Mittwoch nicht empfiehlt Es soll ausschließlich zusätzlich ein Überblick über die Menüs der anderen Tage gegeben werden, ausgewählt werden kann aber nur der aktuelle Tag, in unserem Fall ist dies exemplarisch der Mittwoch.
Wie in der States-Übersicht (Tabelle 1) zu sehen, ist für jedes Ring-Menü der unterschiedlichen Tage ein State erstellt worden. Der Wechsel in einen anderen

State erfolgt über die Verdeckung einer der **Pfeiltasten** (2) und die gleichzeitige Sichtbarkeit des entsprechenden Hauptbildes des Menüs. Auslöser für z.B. State „Donnerstag“ ist die Verdeckung des Markers Pattern2 (Pfeiltaste rechts) und die gleichzeitige Sichtbarkeit des großen Menübildes „Mittwoch“. Um dies zu erreichen mussten vorher der visble-Slot von Pattern2 mit einer NOT-Funktion ausgestattet werden (Verdeckung) und der Ausgang der NOT-Funktion per AND-Funktion mit dem visible-Slot des Menübildes „Mittwoch“ (*Mittwoch_gross*) verbunden werden. Diese AND-Funktion kann nun in der CentralStateComponent als Activator für State01 (Donnerstag) angegeben werden. Auf diese Art und Weise können nun alle Wochentage bzw. die entsprechenden States miteinander verbunden werden.
Genau gleich ist nun auch der **Vegetarisch/Nicht-Vegetarisch-Button** (3) zu integrieren. In diesem Fall ist der visible-Slot von Marker Pattern3 mit einer NOT-Funktion auszustatten, welche wiederum per AND-Funktion mit z.B. dem Bild „Mittwoch_gross“ verbunden wird. Diese wird nun als Activator für State05 benutzt.
Auf zwei Dinge muss noch aufmerksam gemacht werden.
Am Beispiel der Buttons wurden verschiedene Möglichkeiten der Visualisierung getestet. Diese Elemente sind alle über eine BillboardMatrix integriert, wodurch sie der Betrachter immer unabhängig vom Kamerawinkel frontal sieht. Während sowohl der Vegetarisch/Nicht-Vegetarisch-Button, als auch der Getränke/Weiter-Button als sichtbare Elemente (Dependants) direkt in die jeweiligen States integriert wurden, wurde die linke Pfeiltaste direkt und unabhängig von den jeweiligen States visualisiert. Die rechte Pfeiltaste wiederum wurde über ihre Billboard-Matrix indirekt in die States eingebunden. Dies sollte aber nicht nur als Test der verschiedenen Visualisierungsmöglichkeiten dienen, es gibt auch noch zwei weitere Gründe. Zum einen wechseln im Gegensatz zu den Pfeiltasten die erstgenannten Buttons abhängig von den jeweiligen States, der Vegetarisch-Button ist natürlich nur in den States zu sehen, die nichtvegetarische Menüs abbilden, zum anderen sollte getestet werden, welches Verhalten der betätigten Buttons eher den Wünschen des Benutzers entsprechen. Buttons die direkt in einen State integriert wurden, verschwinden nicht bei Verdeckung des entsprechenden Markers. Anders natürlich bei den Pfeiltasten.
Weiterhin gibt es zwei Möglichkeiten der Bewegungsrichtung des Ring-Menüs. Von mir wurden beide Möglichkeiten integriert. Während sich das Getränke-Ring-Menü „in Richtung“ des betätigten Pfeils bewegt, also als wäre das Menü auf einer Scheibe angebracht, die sich in Pfeilrichtung bewegt, ist es bei dem Speisen-Ring-Menü anders herum. Hier erscheint als nächstes Hauptbild dasjenige, welches sich in Richtung des betätigten Pfeils von dem aktuellen Hauptbild befindet.

Kommen wir nun zu dem **Getränke-Button** (4). Theoretisch von jedem vorherigen State aus zu betätigen, funktioniert er in diesem Prototyp nur, wenn das Mittwochsmenü nicht-vegetarisch gewählt wurde. Wie schon erwähnt ist eine Auswahl für jeden Wochentag nicht sinnvoll, ferner ist es für den vorliegenden Entwurf unerheblich, ob nur einer oder aber mehrere Tage ein Betätigen dieses

Buttons erlauben. Die Umsetzung der Funktion des Getränke-Buttons folgt auch wieder dem gleichen System, wie z.B. dem der Pfeiltasten, nur dass in diesem Fall ein Activator für State10 (*Getraenke_Mittwoch*) benötigt wird.

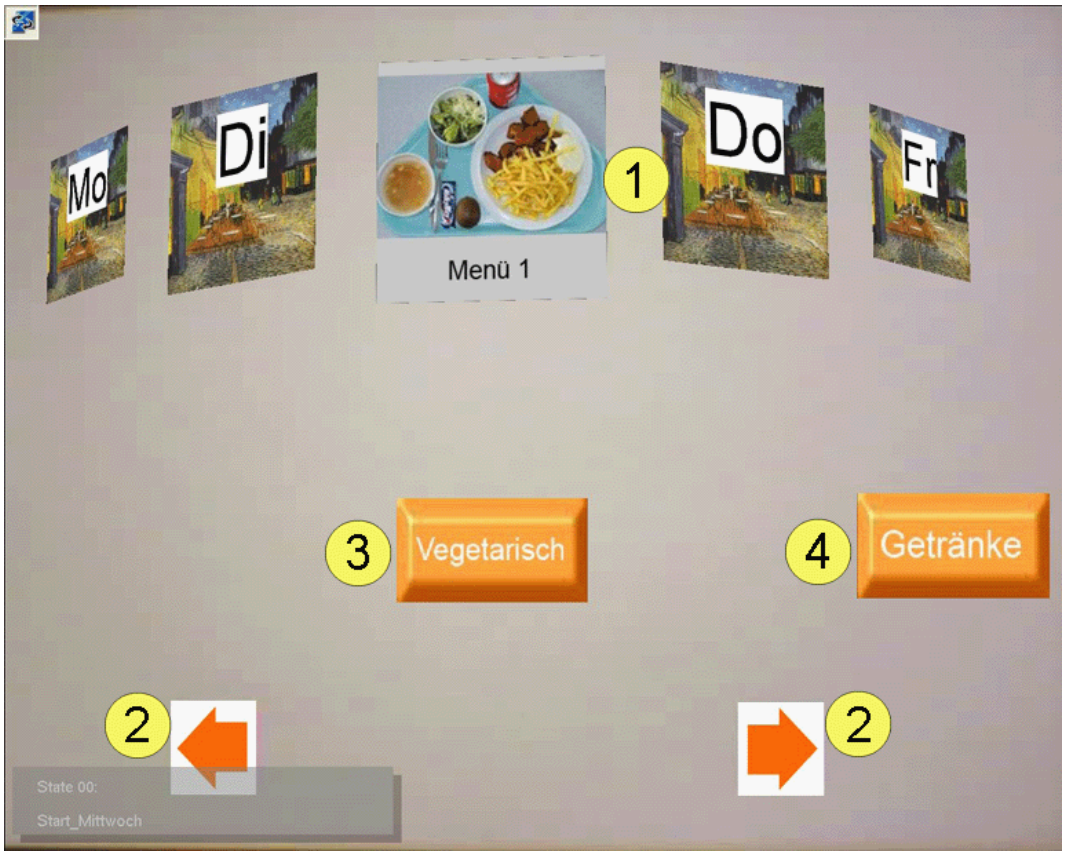

Abbildung 6: Speisen-Menü

In State10 (Abb. 7) angelangt wird nun auf dem Hauptmarker (Pattern Sample1) ein kleineres Ring-Menü für die Getränke (1) eingeblendet. Dies ist auch wieder nach dem gleichen System wie auch schon das Speisen-Ring-Menü aufgebaut. Das mittlere Bild dient als Hauptbild, an welchem sich die kleineren orientieren. Der Zurück-Button (2) fungiert in Kombination mit einem beliebigen Getränke-Hauptbild als Activator für den State01 (Mittwoch). Die Pfeiltasten haben die übliche Funktion.
Exemplarisch ist auch in diesem State die Funktion des **Weiter-Buttons** (3) bei nur einem Getränk aktiviert.

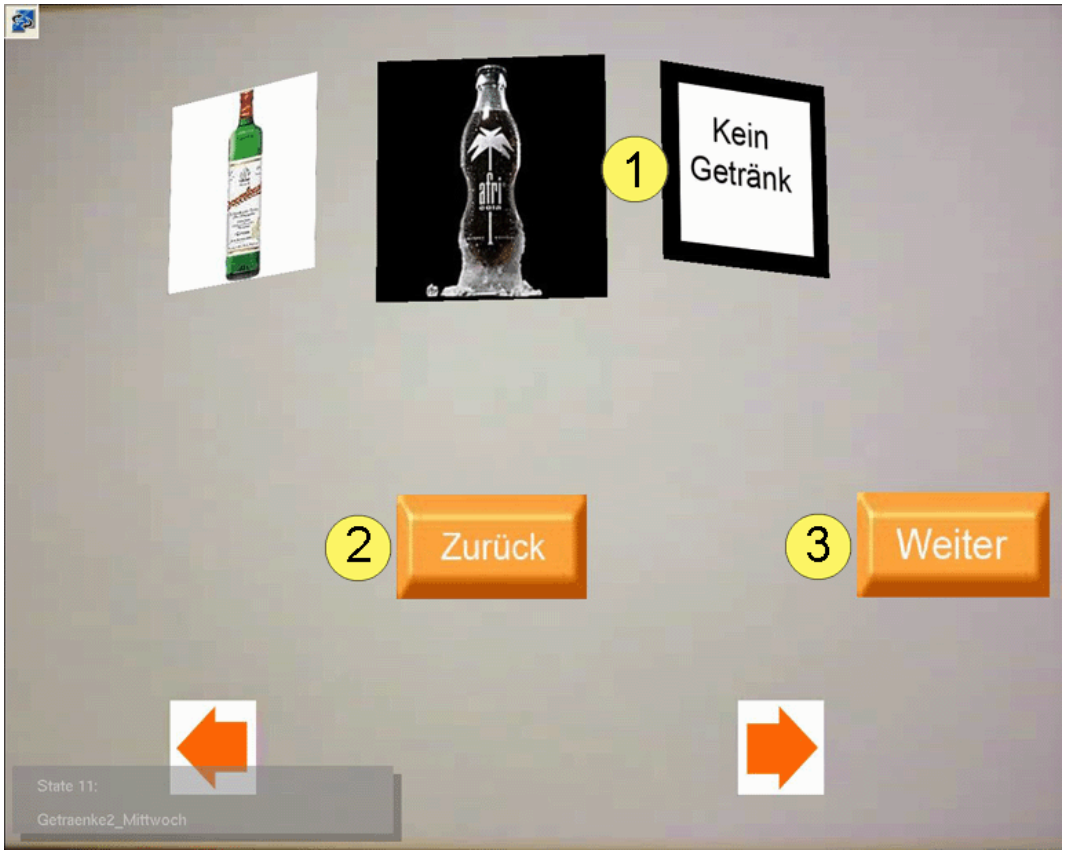

Abbildung 7: Getränke-Menü

Im **vorletzten State** (Abb. 8) angelangt wird noch einmal eine Zusammenfassung der Auswahl auf dem Hauptmarker eingeblendet. Statt des mittleren Buttons erscheint hier jetzt ein Bild der Beilage (1), in diesem Fall Pommes-Frites. Ursprünglich sollte hier eine Wahl des Bräunungsgrades über einen Schieberegler ermöglicht werden. Dieser existiert zwar auch als Element im Objektfenster, machte aber Probleme bei der Integration in die Szene. Ein Regler, wie in der Referenz angegeben, war nicht sichtbar. Um doch noch etwas Ähnliches umzusetzen, integrierte ich eine Auswahl über das LOD-Element (Level Of Detail). Hierfür muss eine Distanz-Funktion mit Marker Pattern3 verbunden werden, so dass die Distanz zwischen Marker und Kamera gemessen wird. Danach wird der distance-Outslot der Distanz-Funktion mit dem distance-Inslot der LOD-Funktion verbunden. Sowohl die Distanz-Funktion, als auch die gerade integrierte LOD-Funktion muss als sichtbares Objekt in diesem State eingefügt werden. Am Ende müssen noch die drei Outslot (showless, show und showmore) mit den visible-Inslots verschiedener Bilder, in diesem Fall dreimal ein Bild der Pommes-Frites mit unterschiedlichem Bräunungsgrad, verbunden werden. Zu erkennen ist die Aktivität der Distanz-Funktion auch durch das eingeblendete Fenster (2) mit den Entfernungs- und Winkelwerten, weshalb selbiges auch absichtlich nicht ausgeblendet wurde.

Abschließend kommt durch Betätigen des **„Kauf ich!“-Buttons** (3) in den letzten State (State14) (Abb.9). Hier wird nur noch eine Abschlussrechnung (1) auf dem Hauptmarker eingeblendet, welche ein Text3D-Objekt ist und einfach als sichtbares Objekt in State14 angegeben wurde.
Zwei **Audio-Komponenten** wurden noch in den Prototyp integriert, eine in State00 mit dem Begrüßungstext, eine in State14, mit der Aufforderung, sich an die Kasse zu begeben. Die Audio-Komponente in State00 wird über eine Delay-Funktion ausgelöst, damit der Benutzer genügend Zeit hat, sich auf das Auswahlmenü zu konzentrieren.

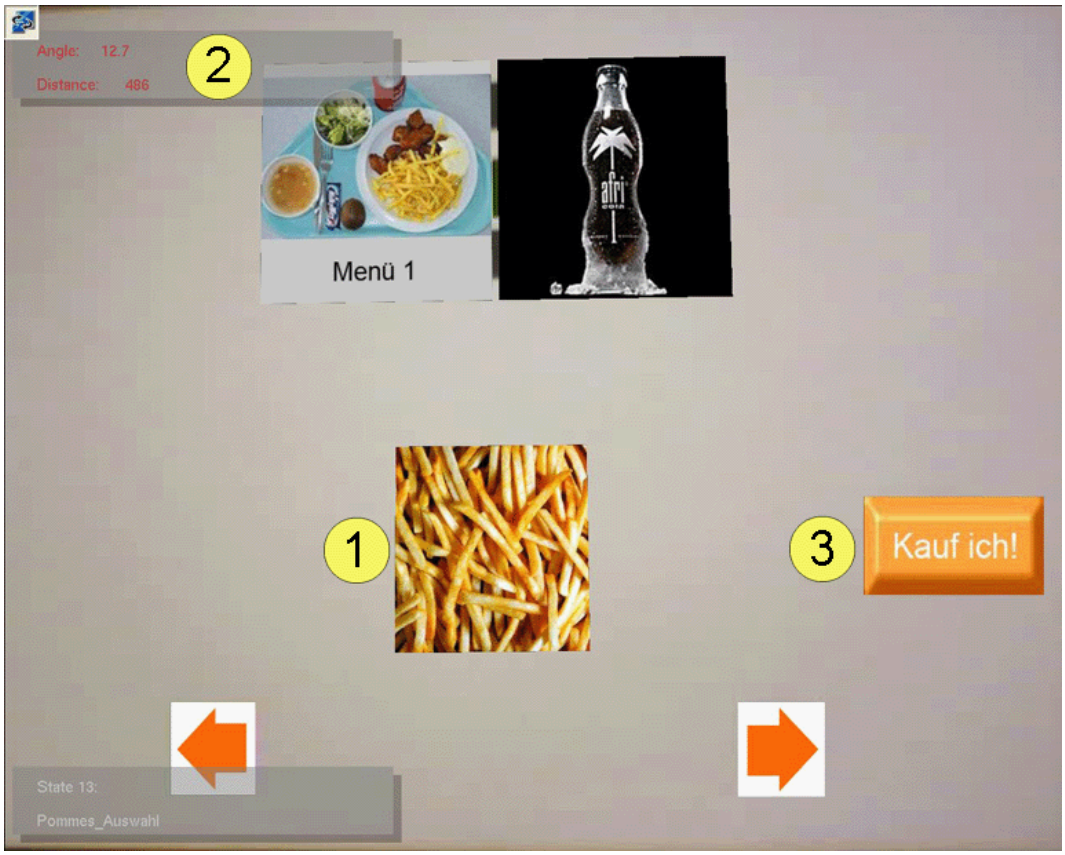

Abbildung 8: Menü-Zusammenfassung

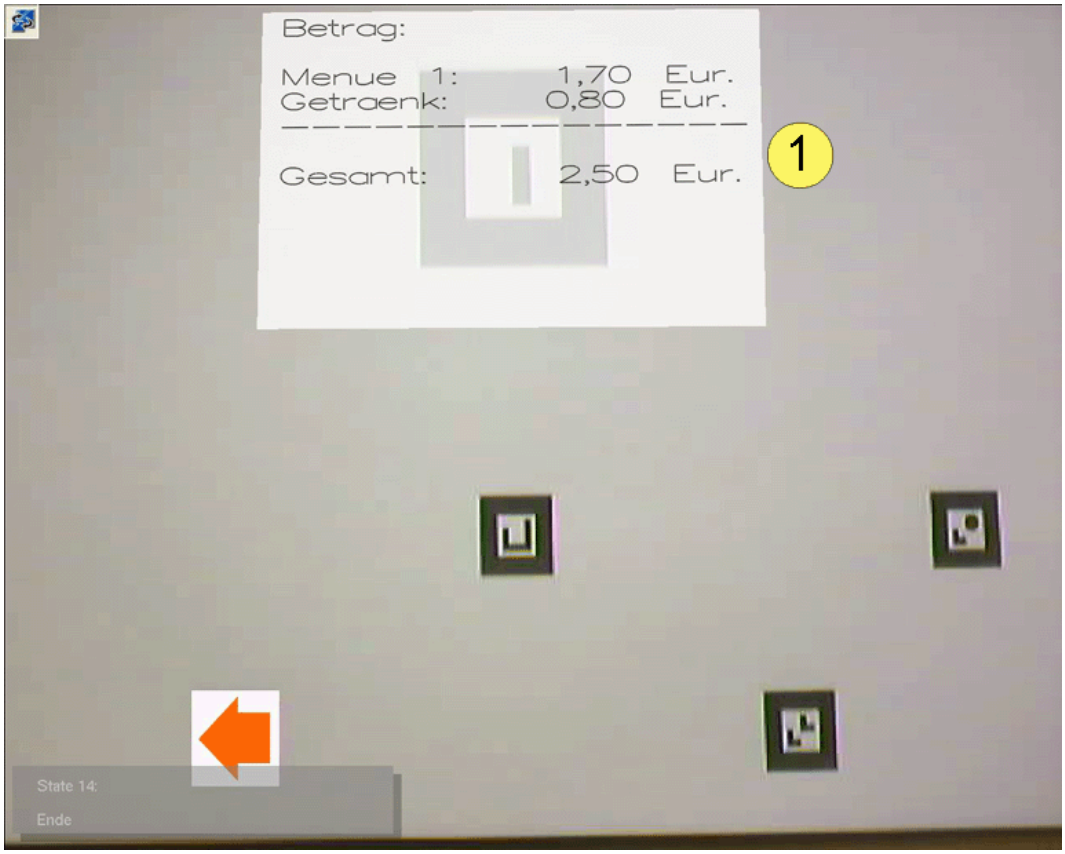

Abbildung 9: Abschluß-Menü

6.5 Probleme und Lösungen

In diesem Abschnitt möchte ich noch kurz auf ein paar Probleme, die der Aufbau des Menüs mit sich brachte, eingehen.
Aus Effizienzgründen wurde z.B. das Ring-Menü auf nur einen Marker abgebildet. Umso weiter aber ein Objekt durch Translation von seinem Marker entfernt wird, desto stärker wirken sich auch Ungenauigkeiten in der Abbildung aus. Schlechte Beleuchtung und die damit verbundenen Probleme beim Tracking haben in diesem Fall zur Folge, dass das Ring-Menü deutlich sichtbar zittert. Das Verwenden von fünf Markern, um jedes Menü-Bild unabhängig von den anderen abzubilden, würde hier Abhilfe schaffen. Dies ist in einer späteren Umsetzung natürlich zu beachten, in unserem Fall für die schnelle Erstellung eines Prototyps aber nicht wichtig.
Bei genauerer Betrachtung des Aufbaus und der Anzahl der States wird schnell klar, dass das Ausführen des Menüs mit seiner vollständigen Funktionalität mit einem erheblichen Aufwand verbunden wäre. Durch den simplen aber dafür schnell zu erstellenden und leicht zu überblickenden Aufbau der Menüstruktur ist es nicht möglich, z.B. die Visualisierung der gewählten Speisen in State13 (gewählte Speisen, Bräunungsgradauswahl der Pommes-Frites) in nur einem State zu verwirklichen, da Unterschiede in der Auswahl nicht individuell an State13 weiter gereicht werden, sondern dies nur durch eine feste Verbindung der einzelnen States möglich ist. Es wurde zwar weiter oben schon erwähnt, dass eine Auswahlmöglichkeit für jeden Tag auch nicht sinnvoll wäre, trotzdem fehlt im vorliegenden Fall noch die Möglichkeit, ein vegetarisches Menü zu wählen, oder sich später zwischen mehreren Getränken zu entscheiden. Aber auch hier muss gesagt werden, dass eine Vergrößerung der Anzahl der Auswahlmöglichkeiten für den grundsätzlichen Aufbau der Menüstruktur und dessen Beurteilung nicht von Interesse ist.
Als nächstes muss noch auf die Technik der Interaktion durch Verdeckung der Marker eingegangen werden. Zwar kann diese Technik simple umgesetzt werden und erfordert auch kein weiteres Nachdenken über Interaktionsmittel und das Tracken (Verfolgen) selbiger, ein unbestreitbarer Nachteil ist aber das unbeabsichtigte Verdecken von Markern. Bei einer zu geringen Menügröße bzw. bei einem zu geringen Abstand der Buttons kann dies leicht passieren. Des Weiteren reagiert das System sehr empfindlich auf Störungen bei mangelhafter Beleuchtung oder kurzfristiger Veränderung der Blickrichtung des Betrachters. Obwohl eigentlich der aktuelle State beibehalten wird, wenn Marker von der Kamera nicht mehr erfasst werden (nur eine Verdeckung bei sichtbarem Marker sollte registriert werden), ist doch eine Störung des Systems zu beobachten.
Ferner musste das Auslösen eines nächstens States durch eine Delay-Funktion verzögert werden. Da eine Verdeckung ein und desselben Markers bei Erscheinen des nächsten Wochentag-Bildes ein Weiterspringen in den darauf folgenden State zur Folge hat, muss die Delay-Funktion für die Zeit, bis die Verdeckung des Markers durch den Benutzer aufgehoben ist, die Sichtbarkeit der entsprechenden Bil-

der des ausgelösten States verzögern. Diese Zeit ist abhängig von der Geschwindigkeit des benutzten Computersystems. Ist also ein vermeintlich wahlloses Umherspringen zwischen den einzelnen States zu beobachten, kann dies meist durch Verbesserung der Sichtverhältnisse oder Heraufsetzen der Verzögerungszeit der Delay-Funktionen behoben werden.
Als letztes ist noch die Frage zu klären, auf welche Art und Weise am besten die Interaktionsobjekte, in diesem Fall die Bilder der Buttons hinter einer Billboard-Matrix, zu integrieren sind. In dem dieser Arbeit beigelegten Prototyp sind verschiedene Möglichkeiten getestet worden.
Grundsätzlich empfiehlt es sich nicht, unabhängig von States Buttons oder sonstige Objekte zu visualisieren. Dies hat, wie in State14 zu sehen ist, zur Folge, dass Bedienungselemente nicht ausgeblendet werden, wenn diese nicht mehr aktiv sind. Werden diese direkt in die jeweiligen States integriert, hat man zwar den Vorteil, dass bei Betätigung der jeweiligen Buttons diese sichtbar bleiben, andererseits aber auch nicht aus dem Sichtfeld gelöscht werden, wenn die entsprechenden Marker nicht detektiert werden. Dies wirkt sich spätestens dann auf den Benutzer irritierend aus, wenn sich keine Marker mehr im Sichtfeld befinden, die Buttons aber an einem Bildrand weiterhin abgebildet werden.
Bewährt hat sich die indirekte Integration in die States, also in zusätzlicher Abhängigkeit von Markern. Dass hierbei der Button bei seiner Betätigung kurz verschwindet, wird weniger als störend, sondern vielmehr als visuelles Feedback, dass das Auslösen erfolgreich war, verstanden.

7 Ergebnisse

Zunächst einmal ist festzustellen, dass das geforderte Testszenario weitgehend in allen Punkten umgesetzt werden konnte. Einzig und allein die Implementierung eines Schiebereglers war nicht möglich. Ob dies aber auf einen Fehler des Programms zurück zu führen ist, oder aber die Gründe anderweitig gesucht werden können, ist abschließend nicht genau zu sagen. Die restlichen Anforderungen aber konnten sehr frei umgesetzt werden, größere Beschränkungen seitens des Frameworks waren während der ganze Phase der Umsetzung nicht zu bemerken.

Die Funktionalität von AMIRE hat bei weitem meine Erwartungen, die ich am Anfang dieser Studienarbeit hatte, übertroffen. Eine zügige Einarbeitung ist ohne weiteres auch für ungeübte Benutzer möglich, detaillierte Kenntnisse in Augmented-Reality nicht zwingend notwendig. Es gibt eine ausreichende Auswahl von Dokumentationen, und allein schon das dem Framework beiliegende *AMIRE-Authoring-Installation and First Steps.pdf* gibt einem auf 37 Seiten einen sehr guten Überblick, bei der Installation angefangen bis hin zu anspruchsvolleren Anwendungen wie das Einfügen der Animations-Komponente. Es darf aber auch nicht verschwiegen werden, dass das Projekt wohl eingestellt wurde, bevor die Dokumentation vervollständigt worden war. Zum einen ist dies daran zu erkennen, dass manche Komponenten in den Dokumentationen nicht mehr mit den vorhandenen im Framework übereinstimmen. Zum anderen ist die in AMIRE integrierte Hilfe nicht mit Inhalten versehen worden. Konkrete Fragen, die ein wenig tiefer in die Materie gehen, sind somit leider nicht immer zu klären.

Sehr positiv aufgefallen ist mir die übersichtliche und ansprechend gestaltete GUI. Man hat zu jeder Zeit einen umfassenden Überblick über die integrierten Komponenten, die Anordnung der Menüs ist klar strukturiert, die Bedienung ist sehr intuitiv. Nicht zu vergessen ist, dass alle durchgeführten Schritte in Echtzeit für den Benutzer sichtbar ausgeführt werden. Dadurch ist jederzeit nachzuvollziehen, ob mit den getätigten Aktionen auch die gewünschten Ergebnisse erzielt werden. Verschiedene Ansätze können problemlos getestet und gegebenenfalls sofort wieder rückgängig gemacht werden.

Abschließend muss gesagt werden, dass ich den Einsatz von AMIRE für die Aufgabenstellung der schnellen Prototypenentwicklung für AR-Anwendungen uneingeschränkt empfehlen kann. Leider ließ es der Zeitrahmen dieser Arbeit nicht zu, genauer auf die große Fülle von Möglichkeiten, die dieses Framework zulässt, einzugehen. Es ist erstaunlich, dass AMIRE in meinem Umfeld der Universität Koblenz-Landau keinen größeren Bekanntheitsgrad erreicht hat. Auch ist es schade, dass die Entwicklung des Programms nicht vollständig abgeschlossen werden konnte.

8 Abbildungsverzeichnis

9 Literaturverzeichnis

LABEIN:
http://www.amire.net, Internetseite des AMIRE-Projekts, (15.01.2007)
http://www.amire.net/publications_deliverables.html, Technische Unterlagen, Analysen, Tutorials etc. des AMIRE-Projekts, (21.01.2007)

GVU-Center, Georgia Institute of Technology:
http://www.gvu.gatech.edu/dart/ , Internetseite des DART-Projekts, (15.01.2007)

Technische Universität München:
http://campar.in.tum.de/Chair/DwarfWhitePaper, Internetseite des Lehr stuhls für Informatikanwendungen in der Medizin & Augmented Reality, (15.01.2007)

Jegust, 2007:
Thomas Jegust, *Analyse und Vergleich von Augmented Reality Frame works aus softwaretechnischer Sich*t, Diplomarbeit Universität Koblenz-Landau, 2007 (erscheint)

Technische Universität Graz:
http://studierstube.icg.tu-graz.ac.at/, Internetseite des Studierstube-Projekts, (20.01.2007)

SourceForge.net:
http://sourceforge.net/projects/amire/, Internetseite der Firma *VA Software*, (20.01.2007)

FH-Hagenberg:
http://webster.fh-hagenberg.at/amire/development.html, Internetseite der FH Oberösterreich, Campus Hagenberg, (21.01.2007)

Müller, 2004:
Prof. Dr. Stefan Müller, *Virtuelle Realität und Augmented Reality*, Untelagen zur Vorlesung im WS04/05, Universität Koblenz-Landau, http://geri.uni-koblenz.de/ws0506/vrarfolien/03_tracking_2.pdf (02.02.2007)