Softwaresysteme für AR
Beispielsysteme, Authoring

Vorlesung „Augmented Reality”

Prof. Dr. Andreas Butz

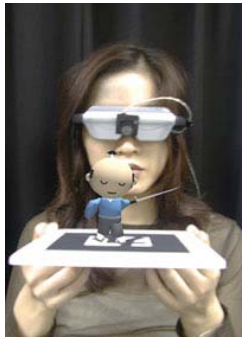
WS 2006/07

Beispielsysteme

- AR Toolkit
- VRIB/ VARIO
- Studierstube
- COTERIE/ Repo-3D
- Tinmith
- DWARF

AR Toolkit

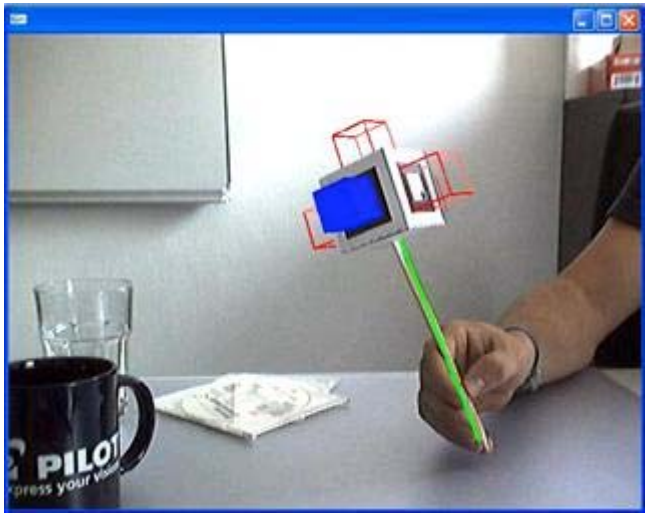
- Entwickelt am HITLab (Seattle, WA, USA)
- C-Bibliothek für Windows, Mac OS X, UNIX
- Bestandteile:
  - Videograbbing (DirectShow, Quicktime, V4L)
  - Markererkennung und -training
  - Hilfsfunktionen für OpenGL-Rendering auf dem Marker
  - Kalibrierungsroutinen



AR Toolkit: Beispielprogramm

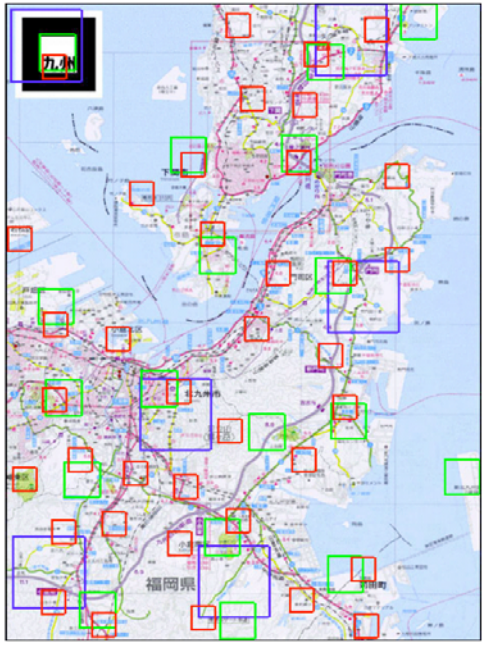
```
arInitCparam(); // init  Kameraparameter
argInit(); // init Ausgabefenster
while (true) {
      arVideoGetImage(); // Bild holen
      argDrawMode2D(); // für Video See-Through:
      argDispImage(); // Bild zeichnen
      arDetectMarker(); // Marker finden
      arGetTransMat(); // Trafo Kamera→Marker finden
      argDrawMode3D();
      argDraw3dCamera();
      glLoadMatrixd(); // Trafo K→M in OpenGL laden
      glutSolidTeapot(); // Teekanne dort zeichnen
}
```

AR Toolkit: Beispiele aus dem Praktikum

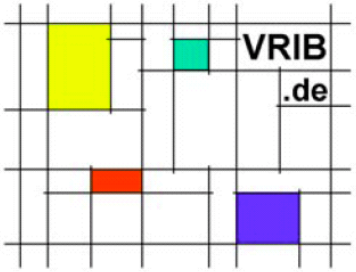


AR Toolkit: Diskussion

- Monolithisches System
- Läuft auf jedem halbwegs aktuellen Rechner (+ billige Webcam)
- Fokus auf markerbasiertem Tracking
  - Rendering nur direkt auf Markern genau
  - Neuentwicklung (soon come…): Texturbasiertes Tracking
- Gut für schnelles Ausprobieren neuer AR-Interaktionen

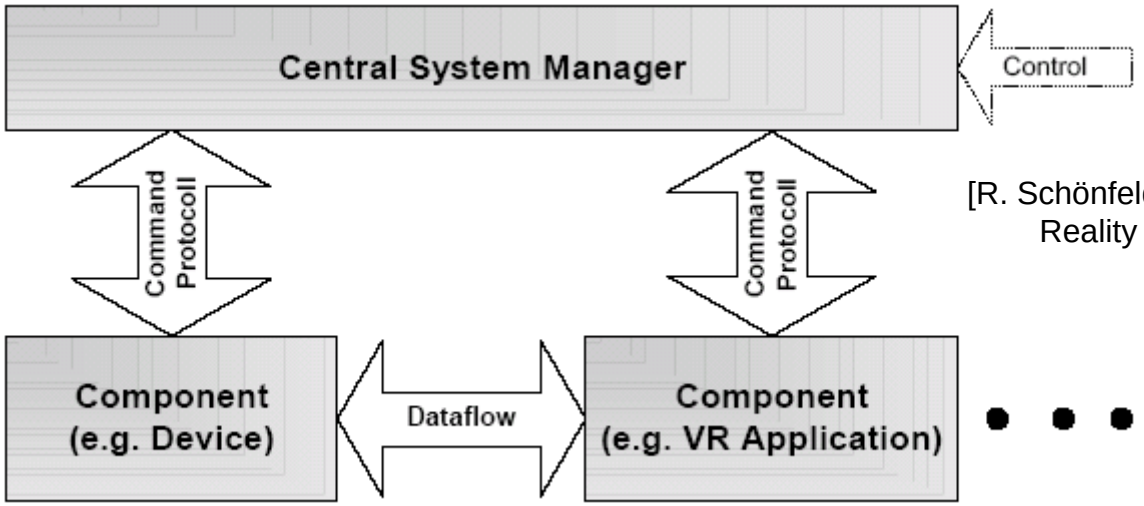


VRIB/ VARIO



- BMBF-Projekt: *Virtual Reality Interaktionsbaukasten* (u.a. TU Ilmenau, DaimlerChrysler R&D)
- Ziel: „*Optimale Unterstützung der Methodik zu Entwurf und Umsetzung von 3D User Interfaces*“, also Rapid Prototyping von 3D User Interfaces
- VARIO (*Virtual and Augmented Reality Inputs and Outputs*) ist verteilte Softwarebasis von VRIB

VARIO: Architektur

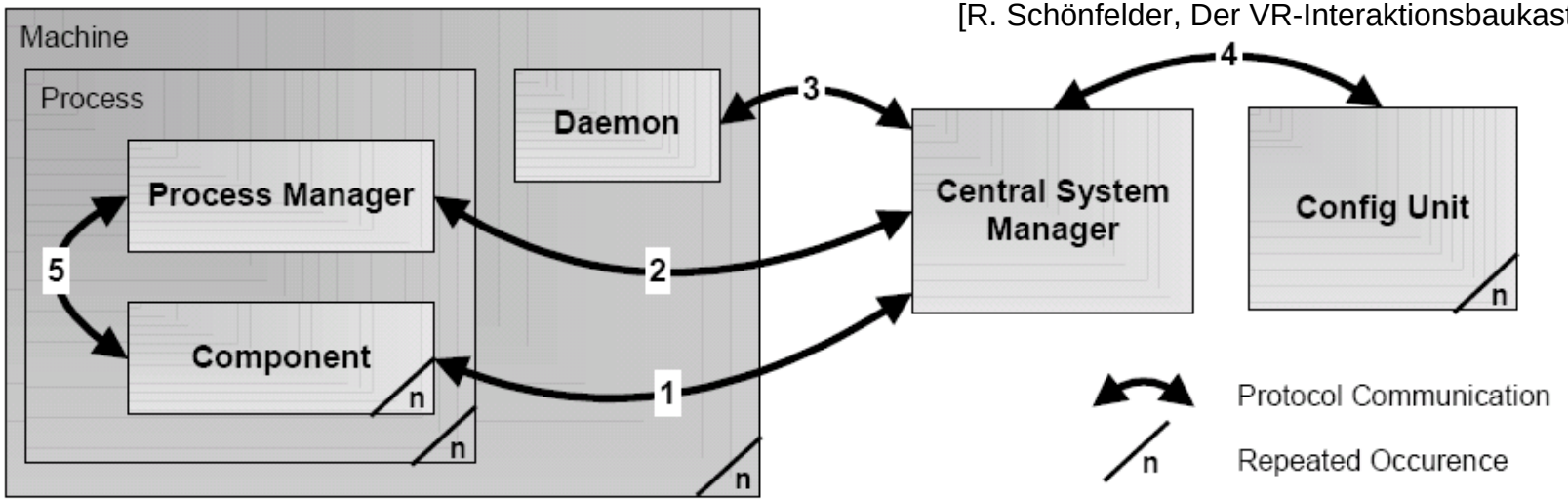


[R. Schönfelder, VRIB – The Virtual Reality Interaction Toolkit]

- Datenflussarchitektur (Pipe and Filter), von Interaktionsgerät hin zur Anwendung
- Zentrale Steuerung durch Central System Manager (CSM)
- Benutzer rekonfiguriert System zur Laufzeit über GUI für CSM
- Persistenz durch Speicherung des Systemzustands im XML-Format

VARIO: Architektur (2)

[R. Schönfelder, Der VR-Interaktionsbaukasten]

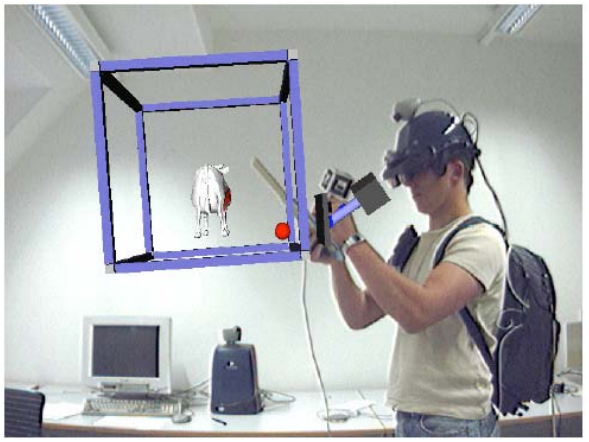


- Auf jedem Host kommuniziert ein *Daemon* mit dem CSM, dieser kann neue *Prozesse* starten
- Ein *Process Manager* kontrolliert je Prozess mehrere Komponenten
- Kommunikation über XML via TCP (Steuerung) und UDP-Streams (Datenfluss)

VARIO: Diskussion

- Zentrale Steuerung erlaubt sehr leichtes Management der Komponenten (GUI)
- Trotzdem hohe Laufzeiteffizienz durch Datenflussarchitektur mit UDP-Streams
- Laufzeitrekonfiguration ermöglicht Rapid Prototyping neuer Interaktionsideen
- Leider nicht öffentlich erhältlich…

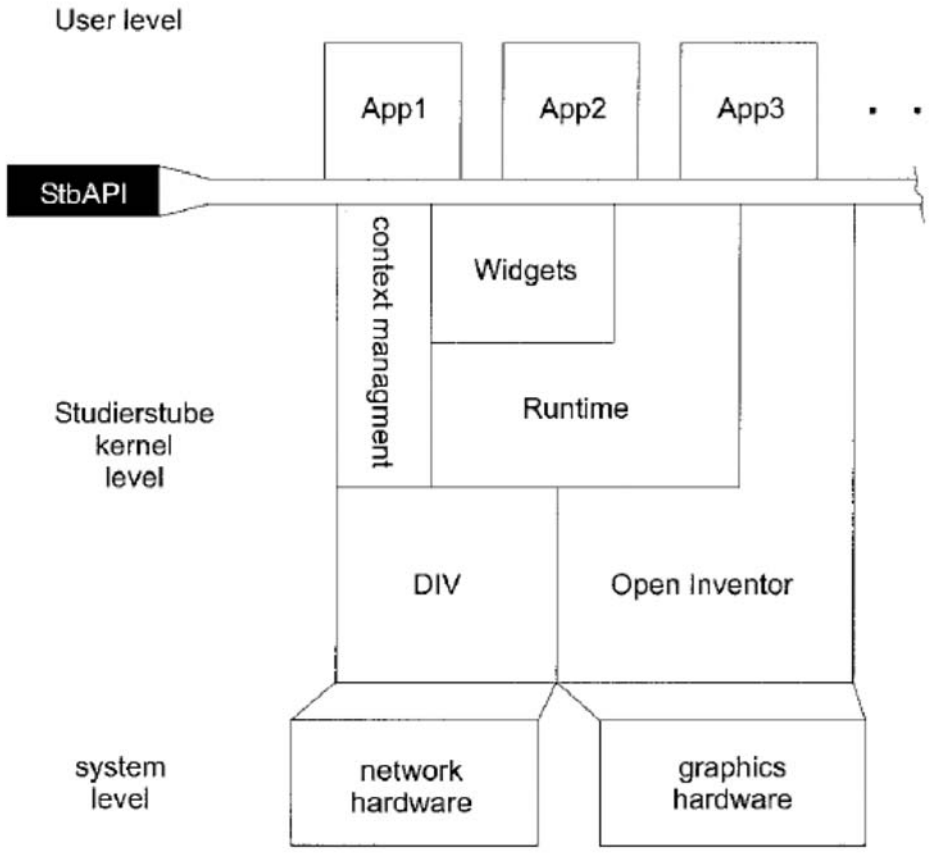
Studierstube



- AR-Framework der TU Wien/Graz (Dieter Schmalstieg)
- Grundlage: Szenengraphbibliothek
- Grundidee:
  - Rendering ist wesentlicher Teil jeder AR-Anwendung
  - Anwendungslogik ist in Knoten des Szenegraphen codiert → wird bei Traversierung des Graphen automatisch ausgeführt

Studierstube: Architektur

- Studierstube Kernel erweitert Open Inventor Szenegraph v.a. durch 3D Widgets
- Distributed Open Inventor (DIV) sorgt für replizierten Szenengraph auf mehreren Maschinen
- Applikationen werden als „3D Windows“ angezeigt; können dynamisch nachgeladen werden
  → Multitasking möglich



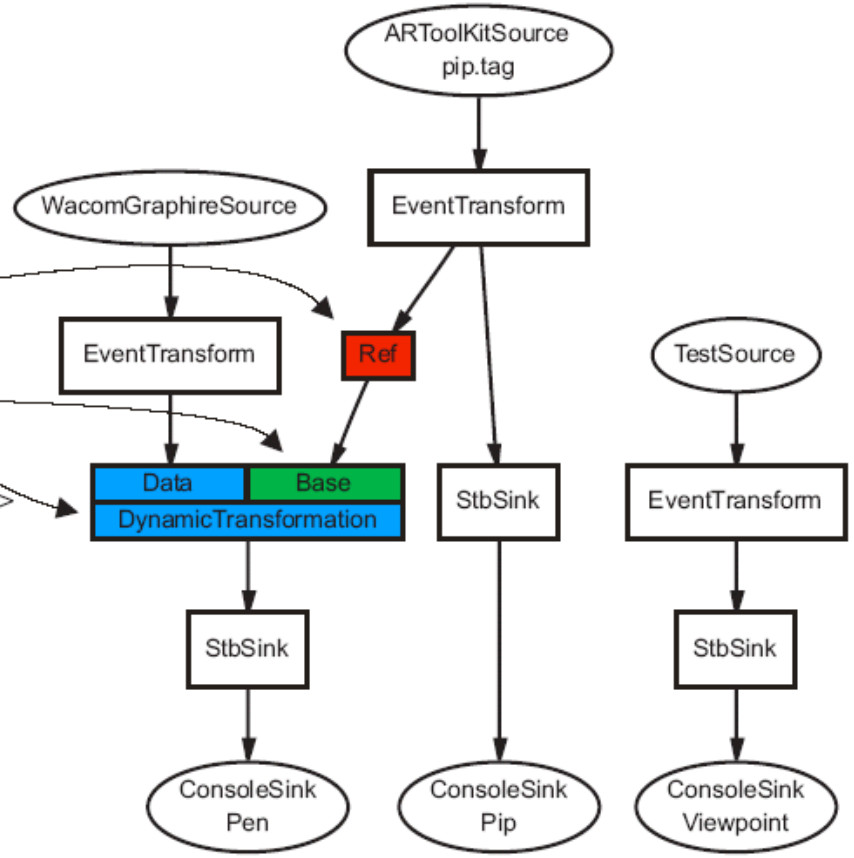
[Schmalstieg et al., The Studierstube Augmented Reality Project]]

Studierstube: OpenTracker

- Problem: wie Tracker anbinden?
  - Erweiterbarkeit, Konfigurierbarkeit
  - Verteilung Rechenbelastung auf mehrere Rechner
  - Daten müssen gefiltert werden (Transformationen, Kombinationen etc.)
- Lösung: Eigenes Framework *OpenTracker*
  - Datenflussarchitektur
  - Source Nodes: Anbindung von Hardware
  - Filter Nodes: Transformationen, Kombinationen
  - Sink Nodes: Abliefern der Daten in Szenengraph
  - Persistente Speicherung in XML-Datei

Stb/OpenTracker: Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OpenTracker SYSTEM "opentracker.dtd">
<OpenTracker>
    <configuration>
        <ARToolKitConfig camera-parameter="camera_para.dat"/>
    </configuration>
    <ConsoleSink comment="Pip">
        <StbSink station="0">
            <EventTransform DEF="Camera" scale="0.001 0.001 0.001">
                <ARToolKitSource tag-file="pip.tag" />
            </EventTransform>
        </StbSink>
    </ConsoleSink>
    <ConsoleSink comment="Pen">
        <StbSink station="1">
            <EventDynamicTransform>
                <TransformBase>
                    <Ref USE="Camera"/>
                </TransformBase>
                <EventTransform scale="-2.1 -2 0" translation="0.14 0.1 -0.01">
                    <WacomGraphireSource device="1"/>
                </EventTransform>
            </EventDynamicTransform>
        </StbSink>
    </ConsoleSink>
    <ConsoleSink comment="Viewpoint">
        <StbSink station="2">
            <EventTransform rotation="1 0 0 0">
                <TestSource frequency="25"/>
            </EventTransform>
        </StbSink>
    </ConsoleSink>
</OpenTracker>
```



[Reitmayr & Schmalstieg, An Open Software Architecture for Virtual Reality Interaction]

Studierstube: Diskussion

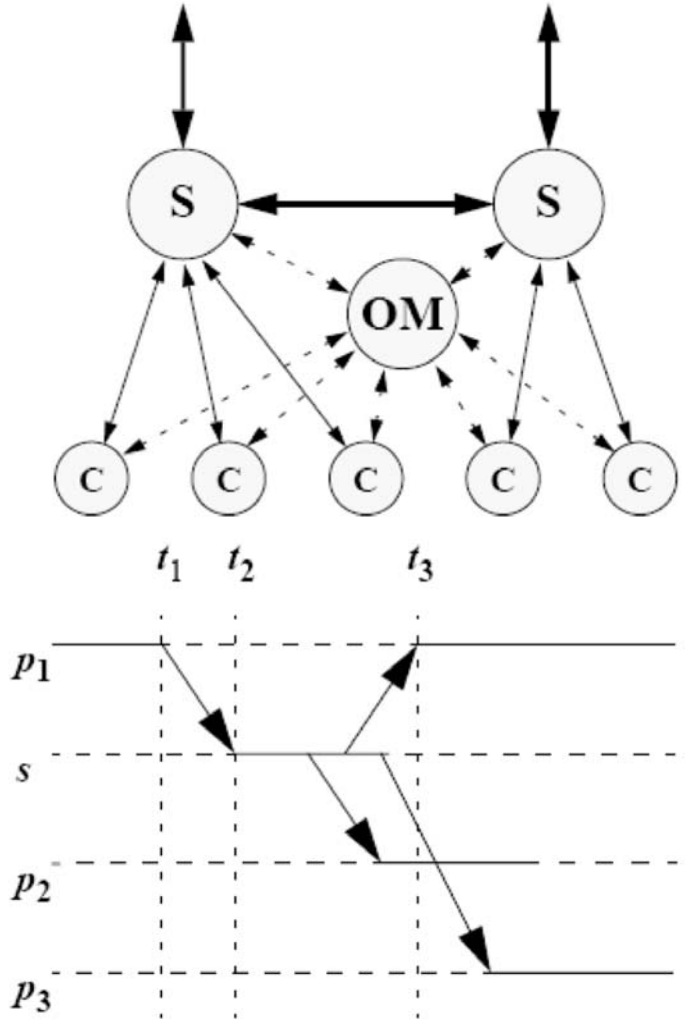
- Starker Fokus auf Interaktionstechniken für 3D-Umgebungen
  - Besondere Betonung des Renderings
- Verteilung:
  - Beim Tracking sehr flexibel, aber statisch
  - Für Anwendungen zur Laufzeit möglich, skaliert allerdings nicht besonders gut (Szenengraph wird voll repliziert)
- Extrem viele Anwendungen und Erweiterungen (u.a. Scripting)
- Frei erhältlich, auch im Source Code

COTERIE/ Repo-3D

- B. MacIntyre, Columbia University, 1996, *„Columbia Object-oriented Testbed for Exploratory Research in Interactive Environments“*
- Cross-Platform Library, Sprache: Modula 3 (Java hat viele der M3-Konzepte übernommen)
- Scriptsprache: Obliq + Repo (Replicated Obliq)
- Grundidee:
  - Verteilter Shared Memory Ansatz (ähnlich Blackboard)
  - Objektorientierung und Multithreading als Grundparadigmen (Kapselung)
- 3D Graphikbibliothek Repo-3D baut auf COTERIE

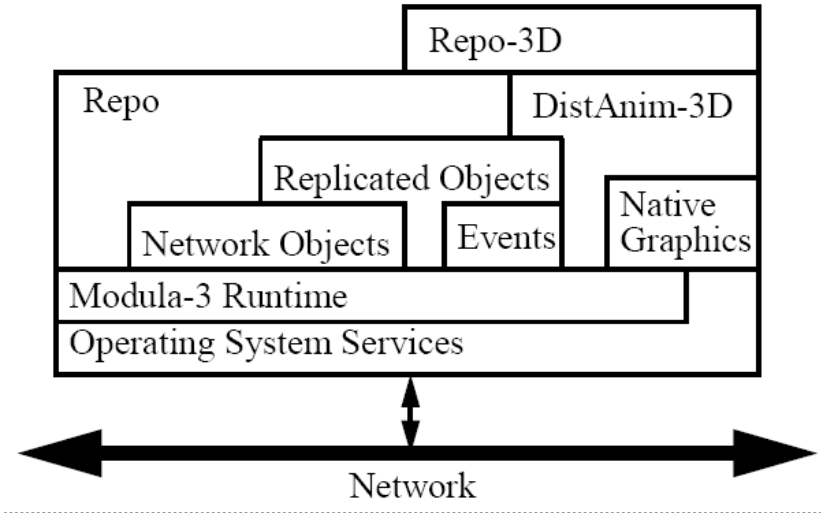
COTERIE: Verteilungsarchitektur

- Hauptproblem: wie stellt man Synchronisation sicher?
- Sequencer empfängt Event, sobald ein Objekt geändert wird
- Dann: Replikation des geänderten Objekts, eventuell auch über weitere Sequencer



[MacIntyre & Feiner, Language-Level Support for Exploratory Programming of Distributed Virtual Environments]

Repo-3D: Architektur



- Repo-3D nutzt COTERIE zur Replikation eines Szenengraphen
- Aber: nicht gesamter Szenengraph wird repliziert, sog. *local variations* (z.B. Kameraposition) sind ausgenommen

COTERIE/ Repo-3D: Diskussion

- System war seiner Zeit weit voraus
- Fokus auf Interaktionstechniken, Rapid Prototyping
- Großer Overhead für Replikation
- Skaliert nur sehr bedingt

Tinmith

- Forschungsplattform, University of South Australia, W. Piekarski
- Fokus auf mobile outdoor AR
- Vermutlich effizientestes AR-System (läuft flüssig mit 40Hz auf 700MHz Pentium III Laptop mit GeForce 2GO)
- Objektorientierte Datenflussarchitektur

Tinmith: Architektur

- Datenflussarchitektur
- Kommunikation zwischen Komponenten über zentralen Object Store
  - Zugriff über Pfade (wie Files), z.B. `/human/body/camera`
  - Serialisierung im XML-Format möglich
  - Durch Serialisierung auch RMI-Mechanismus möglich
- Verteilung über handgestricktes P2P-Netz
- Wichtigste Komponente: Szenengraph

Tinmith: Schichtung & Datenfluss

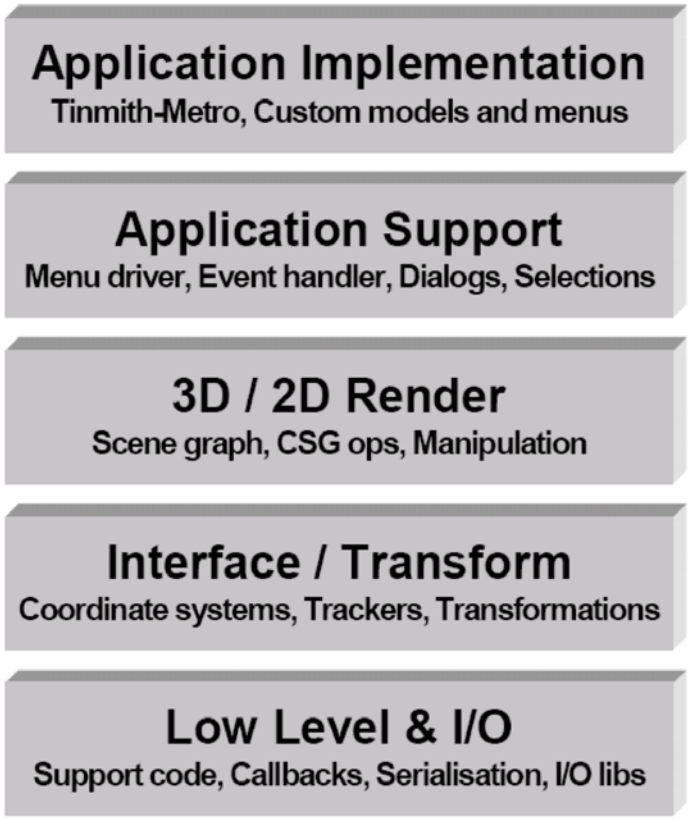


Figure 3 – Layers of libraries with objects available to process data

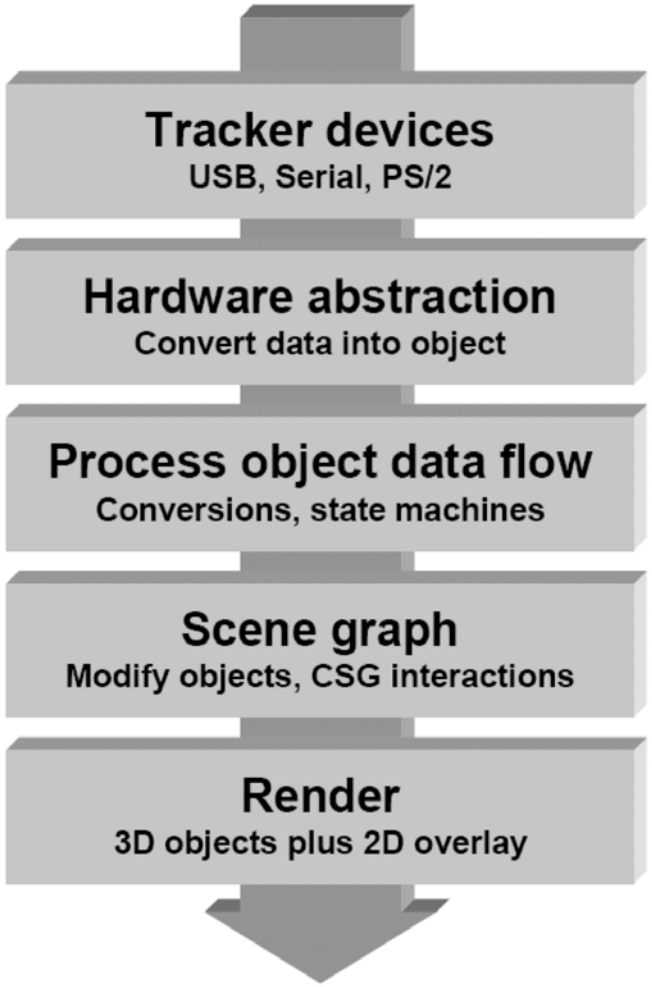


Figure 4 - Expanded view showing stages of processing

[Piekarski & Thomas, An Object-Oriented Software Architecture for 3D Mixed Reality Applications]

Tinmith: Hardwaresystem

- Allgemeines Problem: Wie bringt man viele verschiedene (experimentelle) Hardwaregeräte in einem mobilen Setup unter?
- Teilprobleme:
  - Stromversorgung
  - Interfaces (USB, seriell, Firewire)
  - Trackinggeräte: wo anbringen?
- Tinmith-Lösung: integrierter Rucksack

DWARF

- Distributed Wearable Augmented Reality Framework
  - Zerteiltes Wegtragbares Erweiterte Realität Gerüst
- TU München (AG Prof. Gudrun Klinker), 2001 – heute
- 5 Doktoranden, ca. 30 Diplomanden
- Grundidee: löse Probleme in AR mit Methoden aus dem Software Engineering
- Fokus auf Softwarearchitektur, weniger auf konkreten Anwendungen

DWARF: Design Goals

- Verteilt:
  → flexible, modulare und wieder verwendbare Komponenten als Basis
- Fokus auf AR-Anwendungen
  → effizientes Laufzeitverhalten
- Soll aber auch kleine tragbare Rechner in Ubiquitous Computing Szenarien unterstützen
  → Peer-to-Peer Architektur

→ Ziel: Bringe AR-Interfaces in Ubicomp-Umgebungen

DWARF: Middleware

- Kernideen:
  - Komponenten (*DWARF Services*) als Bausteine
  - Datenflussarchitektur zur Laufzeit
  - Services haben *Needs* und *Abilities*
  - N&A werden durch *Service Manager* verschaltet
  - Kommunikation in zwei Stufen:
    Verbindung über dezentralen Service Manager
    Laufzeitkommunikation direkt zwischen Peers
- Basistechniken:
  - CORBA zur Kommunikation mit dem Service Manager
  - OpenSLP zur Dienstfindung
  - CORBA (RMI), CORBA Notification Service (Events) und Shared Memory als P2P Kommunikationsprotokolle

DWARF: Tracking

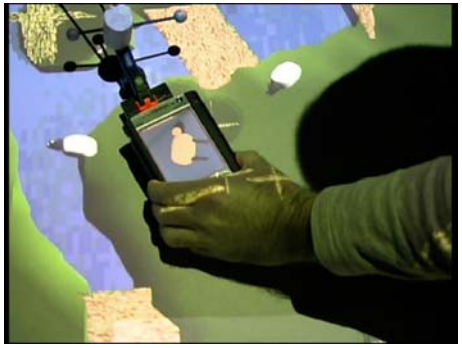
- DWARF Services kapseln COTS (*Custom Off The Shelf*) Hard- und Software (Intersense, ART, GPS NMEA, XSens, AR Toolkit)
- Gemeinsames Datenformat und Koordinatensysteme für alle Trackingdaten
- Verteilungsfeatures der DWARF-Middleware erlauben automatischen Auf- und Umbau komplexer Sensornetze zum Tracking (basierend auf formalem Modell, *Ubiquitous Tracking*)

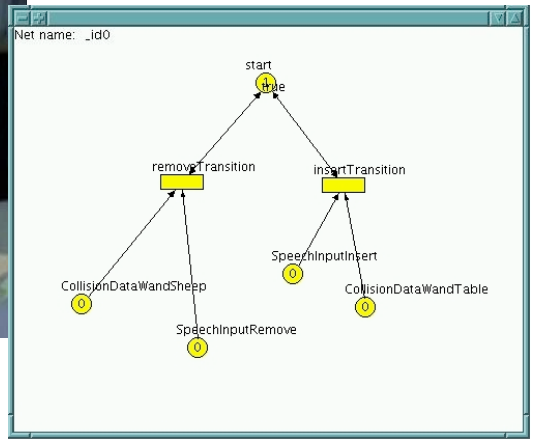
DWARF: User Interface

- User Interface Controller (UIC):
  - Kernidee: Taskflow
  - Modellierung durch Petrinetze
- Fokus auf Verwendung mehrerer Displays: Projektion, getrackte Displays (z.B. Tablet PC), HMD
- 3D-Viewer bietet abstrakte Schnittstelle zur Szenengraphmanipulation, auch Import von VRML/Open Inventor
- Ziel: Programmierung von AR-Interfaces durch Nicht-Experten im AR-Raum

DWARF: Beispiel SHEEP

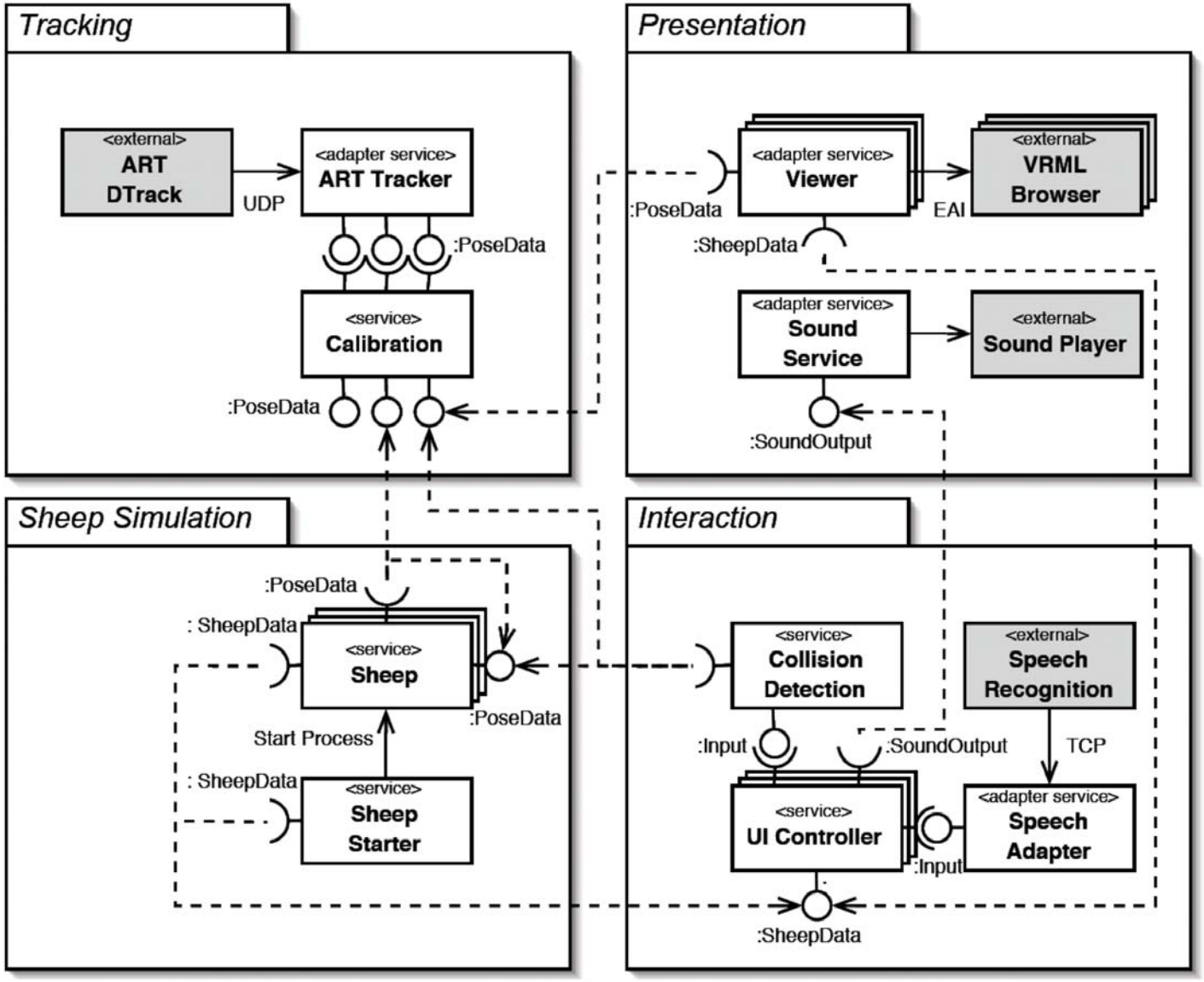
- Shared Environment Entertainment Platform
- Sinnfreies Schaferücken, mit mehreren Interaktionsgeräten und Displays
- Sehr verteilte Implementierung



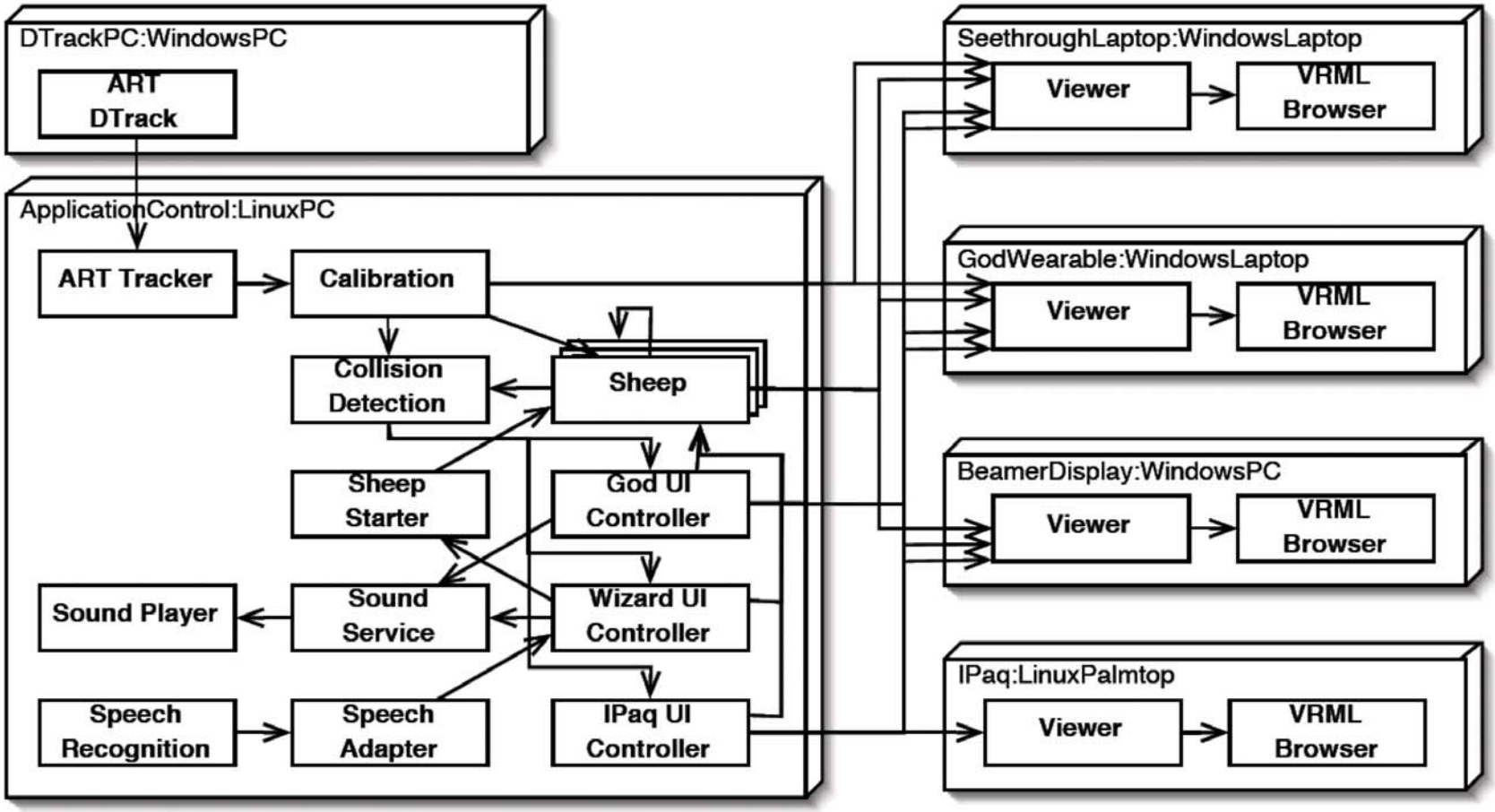


[MacWilliams et al., Herding Sheep: Live System Development for Distributed Augmented Reality]

DWARF: Architektur SHEEP



DWARF: Deployment SHEEP



DWARF: Lessons Learned

- Verteilte AR ist möglich (auch wenn die derzeitige Implementierung besser sein könnte)
- „Echte“ Anwendungen brauchen hohe Flexibilität von DWARF nicht
  - Fokus auf Rapid Prototyping
  - Übernahme Techniken in Endprodukt
- Modellierung eines verteilten Systemzustands ist komplex
  - Zum Glück kommt sowas kaum in Forschungsprototypen vor…

DWARF: Lessons Learned (2)

- Koordination mehrerer Forscher auf einer Softwarebasis ist sehr schwierig
- Zwingende Voraussetzung: klar definiertes gemeinsames Ziel aller Beteiligten
  - Hier: baue ein verteiltes AR-Framework
  - Hier nicht: auf welche Systemteile fokussieren wir uns?
- Zu viel Freiheit im Systemdesign überfordert die meisten Entwickler
  - Templates als Entwurfsvorlagen
  - Einschränkung der Freiheitsgrade

AR Authoring: Herausforderungen

- Programmierung viel zu systemnah, keine klare Aufgabentrennung
- 3D Contenterstellung sehr teuer
- Zusätzliche Sensorik für AR-Systeme zu teuer oder nicht existent
- Entwicklung in der realen Welt sehr teuer (oder gar nicht möglich!)
- Entwicklung in Echtzeit sehr schwierig

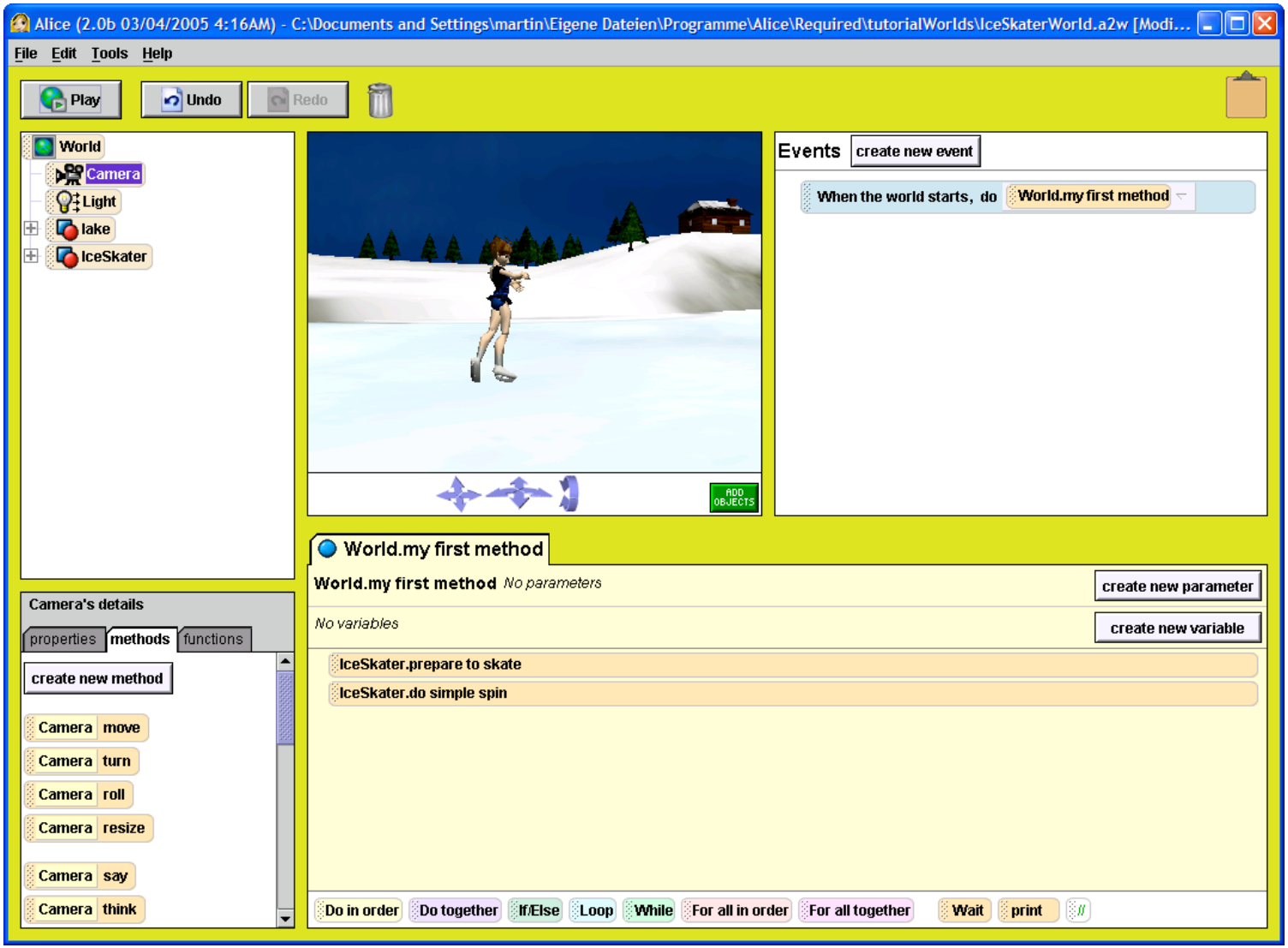
Alice

- Authoring von 3D Animation
- U Virginia, CMU
- Fokus auf Skriptunterstützung zur Animationsmodellierung (Python)
- Umfangreiche Benutzerstudien
  - Intuitive Kommandos („Forward, Up“ statt „x,y“)
  - Kein direkter Kontakt zur grundlegenden Mathematik
- Download unter http://www.alice.org/

Alice: Workflow

1. Szene mit fertigen 3D-Objekten ausstatten (große Bibliothek), Organisation der Objekte in Baumstruktur
2. Skripten des Verhaltens der Objekte
   - Alle Schritte werden direkt animiert
   - Zugriff sowohl über Kommandozeile als auch Kontextmenüs

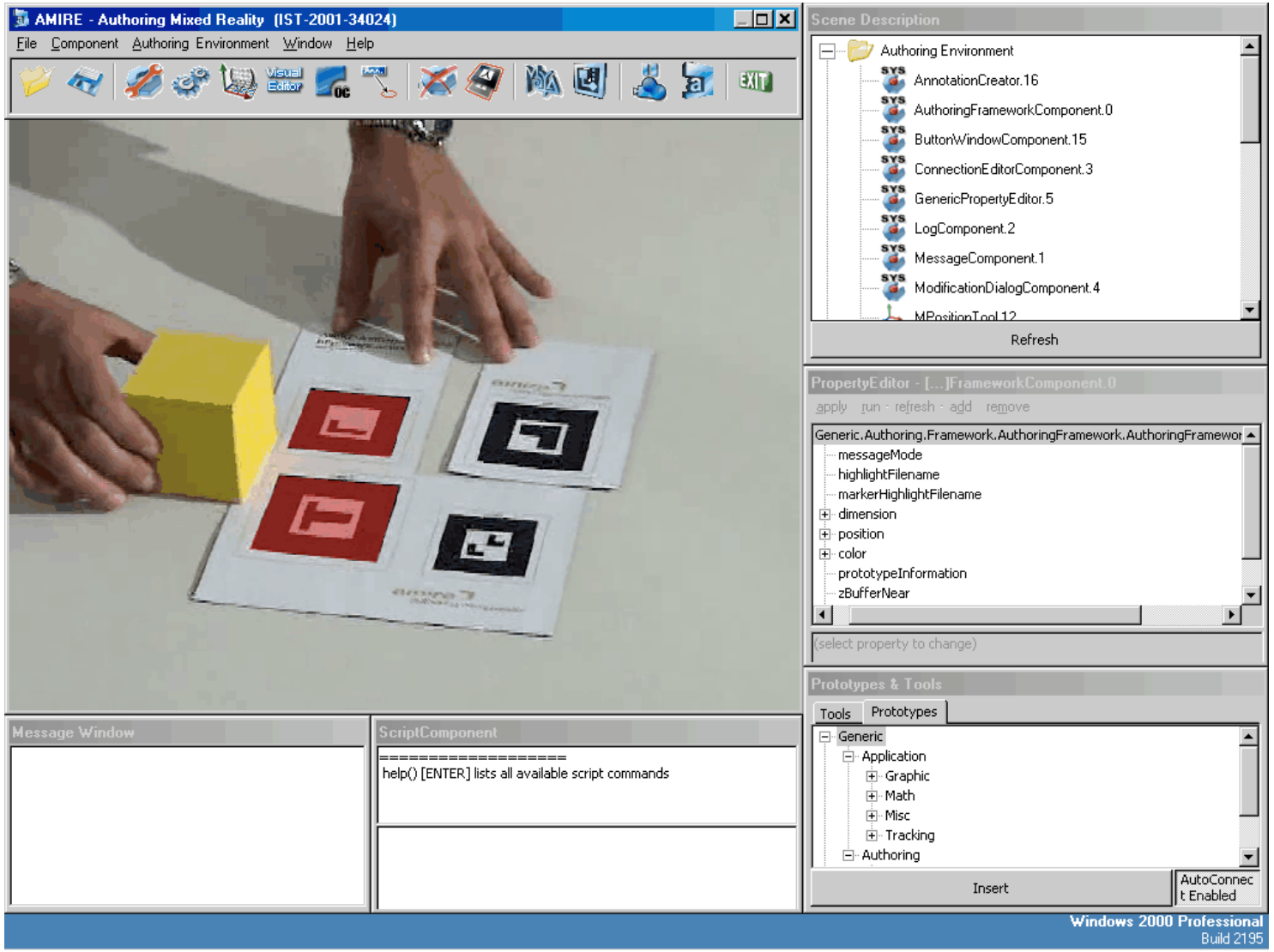
Alice: Beispiel



AMIRE

- EU-Projekt „Authoring Mixed Reality“
- Software-Tools v.a. von der FH Hagenberg bei Linz
- Fokus auf Unterstützung stark strukturierter industrieller Abläufe (Workflows)
- Tracking ausschließlich per AR Toolkit
- Download unter http://sourceforge.net/projects/amire/

AMIRE


AMIRE - Authoring Mixed Reality (IST-2001-34024)
File Component Authoring Environment Window Help
Visual Editor
EXIT
Scene Description
Authoring Environment
AnnotationCreator.16
AuthoringFrameworkComponent.0
ButtonWindowComponent.15
ConnectionEditorComponent.3
GenericPropertyEditor.5
LogComponent.2
MessageComponent.1
ModificationDialogComponent.4
Refresh
PropertyEditor - [...]FrameworkComponent.0
apply run - refresh - add remove
messageMode
highlightFilename
markerHighlightFilename
dimension
position
color
prototypeInformation
zBufferNear
(select property to change)
Prototypes & Tools
Tools
Prototypes
Generic
Application
Graphic
Math
Misc
Tracking
Authoring
Insert
AutoConnect Enabled
Message Window
ScriptComponent
==========
help() [ENTER] lists all available script commands
Windows 2000 Professional
Build 2195

AMIRE: Beispiel Möbelassistent

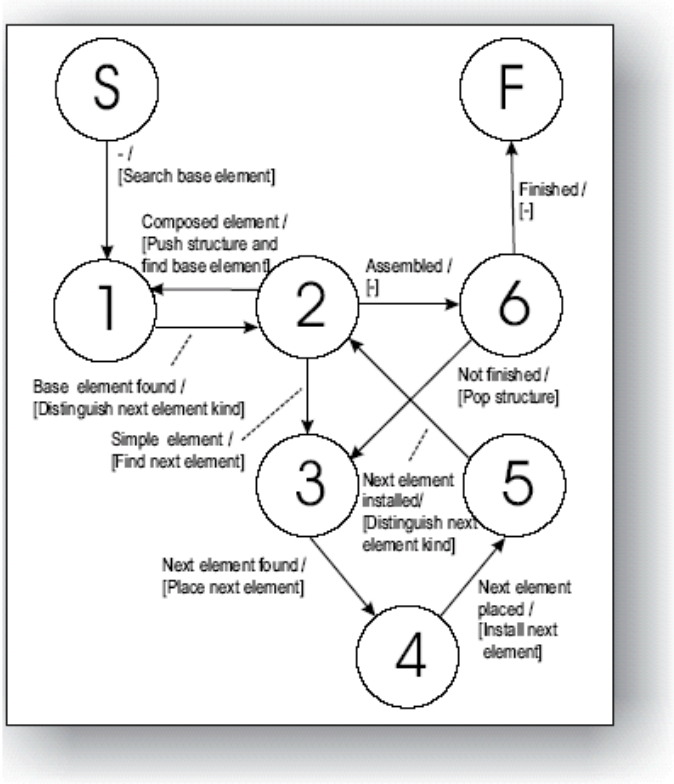


Figure 3. State diagram of the Mixed Reality Assembly Instruction.

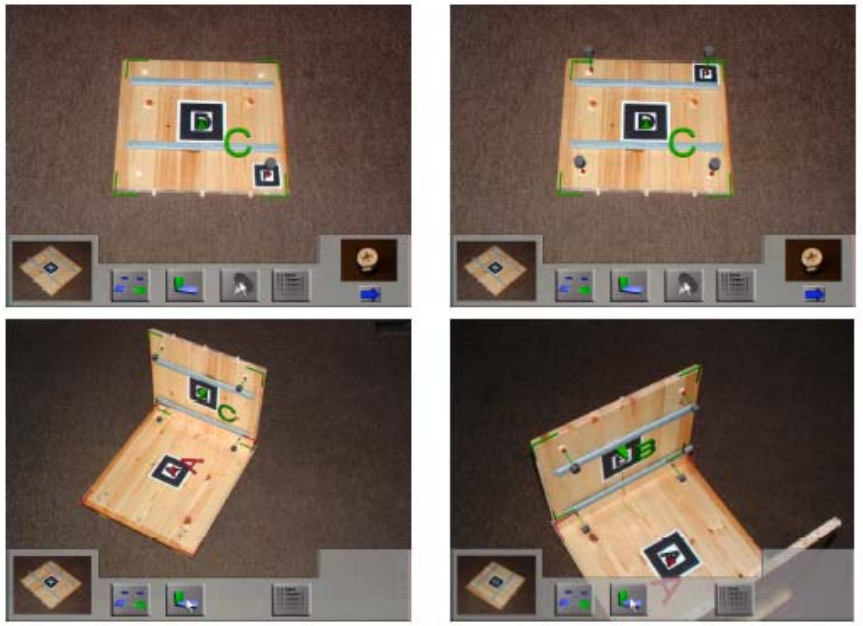


Figure 8. The expert assembles in a step-by-step approach the different parts. The un-detectable and detectable elements are built together with the authoring wizard.

AMIRE: Placement Tool

- Problem: Wie kann man virtuelle Objekte der realen Welt zuordnen?
- Ansatz: Drahtlose 3-Button Maus zum Einstellen von
  - Translation
  - Rotation
  - Skalierung

Figure 15. Placing elements with the placement tool.

DART

- Designer‘s Augmented Reality Toolkit
- Blair MacIntyre, Georgia Institute of Technology
- Basiert auf Macromedia Director, Erweiterung um AR-Komponenten
- Ziel: Unterstützung früher Designphasen, besonders am Übergang vom Storyboard zu Prototypen

DART: Grundlagen Macromedia Director

- *Shockwave 3D* unterstützt dreidimensionale Szenen
- Erweiterbar durch objektorientierte Skriptsprache: *Lingo*
- Elemente
  - Stage: Platz für Content
  - Casts: Speicherung aller Content-Elemente
  - Score: Zentrale Zeitleiste, auf der alles abläuft (normalerweise in Schleife abgespielt)
  - Sprites: Bewegliche Elemente der Casts

DART: Director-Erweiterungen

- Video Capturing
- Marker Tracking (AR Toolkit)
- Externe Trackeranbindung (VRPN)
- Verschiedene Actors (für reale/virtuelle 3D Objekte, Videobilder und Skizzen)
- Unterstützung synchronisierter Aufnahme und Wiedergabe von Audio/Video

DART: Beispiel

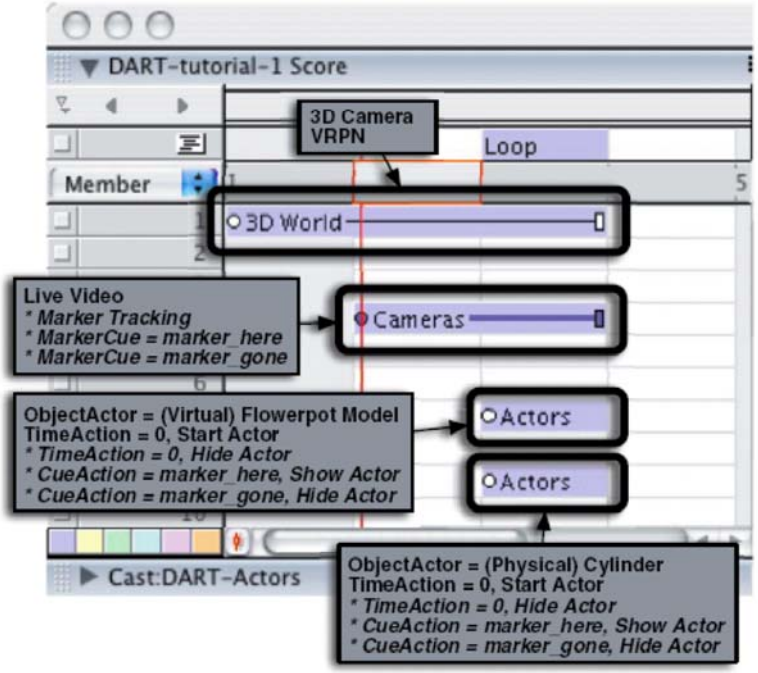


Figure 4: The score for a basic DART example. A screenshot of the running experience is shown in Figure 5. *Loop* is a script that tells Director to repeat this frame forever. *3D World* is a Shockwave3D sprite that covers the stage. *Cameras* and *Actors* are text sprites that are hidden under the *3D World*, and serve as containers for DART behaviors. The behaviors attached to each sprite are listed in the inset boxes, with some key parameters. The italicized behaviors with an asterisk at the start of the line would be included when using marker tracking; if global tracking (via a physical head- or camera-tracker) is used, they are not needed.

Figure 5: A screenshot of a running DART program. The "program" for this example is shown in Figure 4. There are two objects, the virtual flowerpot, and a "physical" cylinder that is aligned with the cup (hard-coded relative to the fiducial in this example) and used to cause the cup to interact correctly with the flowerpot.

[MacIntyre et al., DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences]

Authoring: Diskussion

- AR ist noch sehr junges Forschungsgebiet
- Völlig neue Herausforderungen an Authoring Software
  - Echtzeitprobleme
  - Kosten (3D-Modelle, Sensorik)
- Nur manche Klassen von Anwendungen lassen sich derzeit unterstützen
  - Storyboard-Ansatz (z.B. DART)
  - Wartungsanwendungen (z.B. AMIRE)

Schönes Wochenende (nach der Vorlesungsumfrage).