ARTiFICe – Augmented Reality Framework for Distributed Collaboration

Annette Mossel, Christian Schönauer, Georg Gerstweiler and Hannes Kaufmann

Interactive Media Systems Group, Vienna University of Technology, Austria

***Abstract*—This paper introduces a flexible and powerful software framework based on an off the shelf game engine which is used to develop distributed and collaborative virtual and augmented reality applications. We describe ARTiFICe's flexible design and implementation and demonstrate its use in research and teaching where 97 students in two lab courses developed AR applications with it. Applications are presented on mobile, desktop and immersive systems using low cost 6-DOF input devices (Microsoft Kinect, Razer Hydra, SpaceNavigator), that we integrated into our framework.**

***Index Terms*—3D Interaction Techniques, Mobile Augmented Reality, Low-Cost Tracking Devices, Virtual Reality Framework**

I. INTRODUCTION

Developing Virtual and Augmented Reality (VR/AR) applications requires a lightweight and flexible but still powerful VR/AR framework, which is extendable to easily integrate new devices and technologies. A wide variety of VR/AR hardware and software setups have been built in the past. However, all share a common general system architecture.

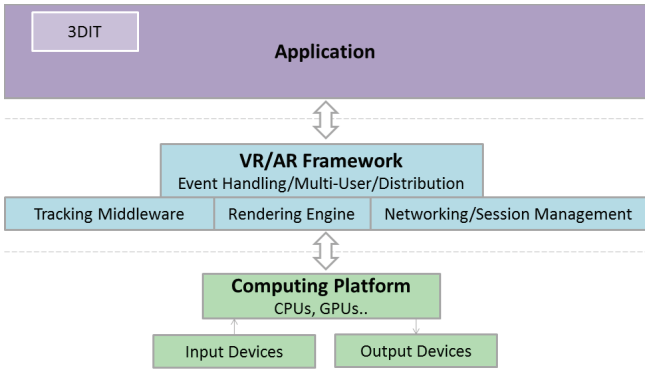

Fig. 1. VR/AR System Architecture

The key elements of an VR/AR system are shown in Fig. 1 and comprise input and output devices whose spatial position and orientation might be tracked, a computing platform with a powerful graphics processor and a VR/AR software framework handling input, output and application behaviour. The most important part is the user (or multiple, collaborating users) working on a certain task and interacting with the system.

Diverse technologies for tracking are in use to determine the location of input and output devices as well as specific body parts of the user up to full body motion capture. Tracking data of these devices is received by the computing platform (e.g. workstation, mobile device) and handed over to the VR/AR framework's tracking middleware. The middleware processes and transforms input data to provide it for subsequent usage within the application. Based on this input data 3D interaction techniques can be supplied to the user. Recent low-cost video game controllers as well as powerful mobile hardware offer great opportunities for novel interaction supporting multiple users for collaborative virtual reality applications.

If multiple users work together, communication is controlled by a network layer while 3D interaction is handled by an event handling mechanism. Subsequently, the virtual scene is visualized to the user on its output device using the rendering engine. Ideally, a VR/AR framework should offer high quality real-time rendering, physics support, networking and scene management to build rich 3D applications. Additionally, for research and teaching purposes, a virtual reality framework must be inexpensive, quick to familiarize with, well documented and flexible for feature extension and rapid integration of novel hardware solutions.

Existing toolkits and approaches, described in section II, have various drawbacks regarding costs, usability, flexibility and extensibility. Thus, we decided to develop a framework based on an off the shelf game engine for collaborative and distributed VR/AR applications supporting multiple users and various input devices for interaction.

Overall our framework provides the following features: (1) a graphical user interface and scene management for rapid prototyping of a VR/AR application, (2) an adaptable interaction and distribution framework for collaborative applications for mobile as well as workstation-based VR/AR setups, and (3) it supports versatile VR/AR setups (mobile, semi-immersive, immersive) on different operating systems and platforms, integrates various input devices such as 2D markers, 3D mice, video game controllers, depth cameras, 6 DOF targets, and supports a range of output devices e.g. smartphones, tablets, stereo projectors, head mounted displays (HMD).

In this paper we present three main contributions: (1) development of an innovative interaction framework for mobile as well as workstation-based VR/AR setups, (2) design and implementation of a complete, flexible middleware/tracking framework for straight forward integration of new interaction devices, and (3) integration of new, low-cost controllers like Microsoft Kinect, Razer Hydra, Sony Move and SpaceNavigator.

Manuscript Received on March 1,2012
mossel@ims.tuwien.ac.at

II. Related Work

Since the mid-1990s, a large number of VR/AR frameworks have been developed and a variety of systems supporting distributed VR applications emerged [5]. Most software frameworks are based on scene graph libraries, e.g., open source toolkits such as Studierstube [16], VR Juggler [3], Avango [7] or commercial ones like 3DVIA Virtools [22]. They provide varying support of multiple input and output devices. In the following, we briefly discuss frameworks related to our work.

Studierstube is an application framework for collaborative Augmented Reality. Its development started in 1996 and continued for almost ten years while it was used for research and teaching. It simultaneously supports multiple users as well as multiple applications, which are embedded as nodes in a scene graph. While this open source C++ based framework is very powerful, it is hard to maintain, does not provide a graphical user interface for scene management and is difficult to learn within a short period of time, which is important if used for teaching. Additionally, recent technologies such as mobile phones or depth imaging sensors can hardly be integrated.

3DVIA Virtools is a commercial development and deployment platform for interactive 3D content creation. It supports multiple users and physics behaviour to create immersive and distributed applications using industry standard VR peripherals. It offers a comprehensive graphical development environment and can deploy to a wide range of output devices. However, its application for research and teaching is limited due to high licensing costs.

One of the first AR frameworks using off the shelf software to design and develop AR applications was DART [8]. DART is based on the Macromedia Director multimedia programming environment. It uses the familiar Director paradigms of a score, sprites and behaviours to allow a user to visually create complex AR applications. DART also provides low-level support for the management of trackers, sensors, and cameras via a Director plug-in Xtra. However, DART is not suitable for research and teaching due to licencing costs for Director. In addition, it lacks stereo output support and the timeline based scene management is rather made for story telling environments than VR/AR applications.

Similar to Virtools, Unity3D [17] features an editor for authoring 2D and 3D content and compromises a game engine for executing the application. Nevertheless, Unity3D by itself is no VR/AR framework but is designed for creating 3D video games and other interactive content. It offers a powerful render engine providing lighting, physics, network communication for collaboration and content distribution. Furthermore, it provides an integrated programming environment using C#, JavaScript or Boo while development can be done under Windows as well as Mac OS X. The final application can be built – generally without changes – for various platforms such as Windows, Mac, iOS, Android, all major game consoles, Flash and web clients. Unity3D is available for free and applications can be deployed at no charge to Windows and Mac.

Our framework uses Unity3D as its underlying development platform and rendering engine. All VR/AR specific extensions were built around Unity3D.

III. Framework Overview & Data Flow

Concerning our motivation stated in section I we aimed on developing a loosely coupled modular software framework which can easily be adapted to support novel devices and interaction techniques. In Fig. 2, an overview of the framework with its components and the data flow is illustrated.

Tracking data from various workstation-based input devices as well as mobile devices are fed into the framework using a transparent and adaptive middleware layer. The middleware transforms all input data in a consistent way and delivers it to the application layer. The application layer is built on top of an external game engine. Within the application layer, the ARTiFICe core handles the tracking input data, provides interaction technique and distribution support and delivers the data to the game engine's scene management. The virtual scene with real-time interaction is then visualized on different output devices using the game engine's rendering module.

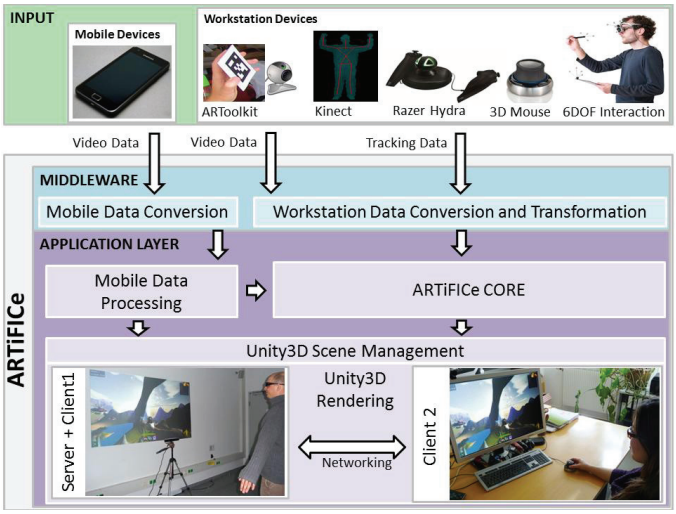

Fig. 2. Framework components and data flow

To provide various 2D as well as 3D interaction techniques the ARTiFICe core comprises a flexible interaction framework, which offers multiple pre-defined interaction techniques and is easy to extend with novel interaction metaphors. The interaction framework offers a single interface to process tracking data from a workstation based device or from a mobile phone to control the virtual interaction device for selecting and manipulating virtual content.

Besides interaction with 3D content, the co-presence of multiple users interacting with the same content at the same point in time opens up great possibilities for collaborative work. Hence, we integrated a distribution framework into the ARTiFICe core to enable real time user-managed collaboration for various hardware setups for two or more users over the network. Further details about the ARTiFICe core are given in the next section.

IV. Implementation

We decided to use Unity3D as the base infrastructure for the application layer of our proposed VR/AR framework. It offers a powerful rendering engine, the possibility to extend its functionality with own features and a strong developer community. Furthermore, the free to use license which includes

all features we need for authoring, rendering and physics support. All implementations are done using C++ and C#.

4.1 Middleware Layer

To gather all tracking input by various devices in one single software layer we use OpenTracker (OT) [15]. It is open source tracking middleware, which offers transparent and flexible device integration, pre-processes input events and passes them to the application layer. It provides a framework for the different tasks involved in tracking input devices in VR/AR applications and eases the development and maintenance of hardware setups in a flexible manner. This is achieved by using an object-oriented design based on XML and utilizing standard XML tools for development, configuration and documentation. A multi-threaded execution model takes care of tunable performance, filters and transformations can be applied to tracking data. XML based configuration files are used to describe tracking configurations that usually consist of multiple input devices. To fetch tracking data from remote input devices, Virtual-Reality Private Network (VRPN) [18] can be used. It is a device-independent and network-transparent framework for devices used in VR/AR systems.

Hence, to provide a transparent interface and for loose coupling between the set of physical devices and the application layer we integrated OT as middleware in ARTiFICe to hand over the tracking data from input devices. Therefore, we developed a new OT node called “Unity” to provide a single sink for all tracking devices. The Unity-node is referenced during run-time by the ARTiFICe core for fetching tracking data to provide them within the application. To access tracking data over the network in OT we use VRPN interfaces.

4.2 Tracking Devices & Integration

We integrated support for several tracking systems in our framework to enable desktop-based, semi-immersive, full immersive as well as mobile VR/AR setups.

For desktop setups ARToolKit [6] as well as ARToolkit+ [21] are easy to use tracking libraries providing a square planar shape for pose estimation and an embedded 2D pattern for distinguishing markers. They calculate camera position and orientation relative to physical markers in real time and thereby enable the development of a wide range of Augmented Reality applications. ARToolkit is usually applied for desktop based AR environments while ARToolkit+ enhances the original ARToolkit library and is optimized for usage on mobile devices. We use ARToolkit+ within our framework, which has been previously integrated into OT. OpenVideo [10], a data integration- and processing framework, is applied to acquire video frames from the webcam, which are processed by ARToolkit+ and later streamed into Unity3D to provide a view of the real world scene.

We integrated a 3D mouse and Razer Hydra into the OT Unity-node to enable 6DOF desktop interaction. For semi- as well as fully immersive mixed reality applications iotracker [13] as a passive marker based infrared optical tracking system is suitable. It tracks arbitrary physical objects furnished with passive markers with an update rate of 60 Hz and very low latency (20-40ms), minimal jitter (RMS less than 0.05mm), submillimeter location resolution and an absolute accuracy of ±0.5cm. For integration we interfaced OT with iotracker using VRPN and the OT Unity-node.

Besides marker-based optical tracking we also wanted to allow markerless full-body motion tracking. Therefore, we integrated Microsoft Kinect using the OpenNI/NITE [9][12] framework and FAAST [17]. OpenNI/NITE provides an API to access raw depth data as well skeleton data, which are calculated based on the depth data. FAAST runs as self-contained application and reads this data. It provides gesture recognition support and full body tracking data via VRPN to OT. Using the OT Unity-node we feed real-time skeleton tracking and gestures into the ARTiFICe core.

The development of AR applications for current handheld devices must also be supported by a modern VR/AR framework. Vuforia [20] is a tracking framework that enables creation of mobile AR applications by using the display of the mobile device as a "magic lens" into an augmented world. It detects a variety of real 2D and 3D objects by robust natural feature tracking and hands over tracking- and pose-information to the AR application to visualize artificial content within the live camera image in the smartphone display. It runs on iOS and Android. In our framework Vuforia acts as mobile middleware and mobile data processing component and is interfaced to the ARTiFICe core.

Our flexible middleware concept allows configuration of all devices and any combination of them using a single OT XML configuration file. Configuration of mobile devices is treated separately using Vuforia.

4.3 Application Layer

In Fig. 3, a detailed view on our framework with its data flow and core components is given. The ARTiFICe core comprises a Manager and a tracking-, interaction- as well as distribution framework.

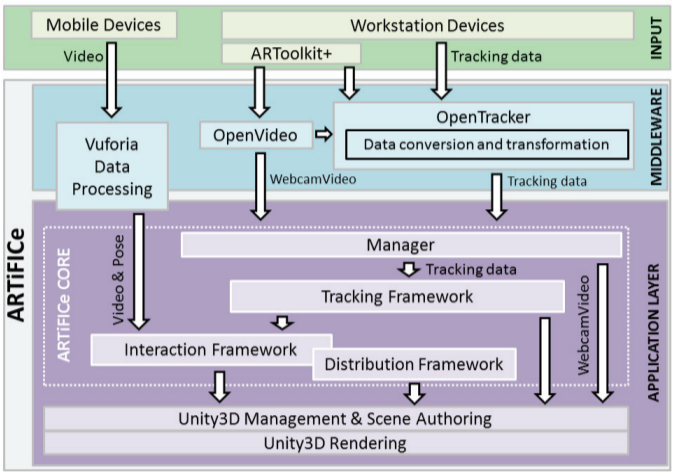

Fig. 3. Detailed framework components

The ARTiFICe Manager controls the data flow between middleware and application layer. At application start-up it reads the OpenVideo and OpenTracker configuration files and loads the dependent tracking libraries. It starts an OpenTracker instance and an OpenVideo handler for ARToolkit+ marker tracking. It also closes OpenVideo and stops OpenTracker at application shutdown. A detailed explanation of the tracking-,

interaction- and distribution framework is given in the next subsections.

All virtual objects in the scene are managed by Unity3D's scene authoring. Therefore, Unity3D provides a C# container class called *GameObject* to which geometry, transformation nodes, textures, physical properties as well as own C# classes can be attached to control visual appearance and overall behaviour of a virtual scene object. These *GameObjects* can be grouped in a logical manner forming a transformation hierarchy, which corresponds to the matrix stack concept in OpenGL. For in depth explanations please refer to the Unity3D documentation [19].

1) Tracking Framework:

To feed tracking data of an input device into the transformation node of a *GameObject* for mapping the real physical position and orientation to a virtual object we developed the ARTiFICe tracking framework. It inherits from the Unity3D base class *MonoBehaviour* to be able to attach the deriving classes to any virtual scene object. The overall design of the tracking framework is shown in Fig. 4.

The tracking framework differs between mobile and workstation-based tracking input. In the mobile setup *TrackMobile* accesses the current device pose by using *Vuforia.TrackerBehaviour*. For the various workstation setups a subclass for each supported device was implemented. Depending on the input device the corresponding subclass is attached to the virtual scene object. As soon as the application starts *TrackProvide*r creates ARTiFICe Trackers through the ARTiFICe Manager, which is implemented as singleton. Each ARTiFICe Tracker is interfaced to the corresponding OT Unity-node and thereby provides the input data of all used devices to the tracking framework. In addition, for 2D marker tracking we developed our own multi-marker tracking support to be able to track cuboid-formed 3D objects and determine its absolute physical pose.

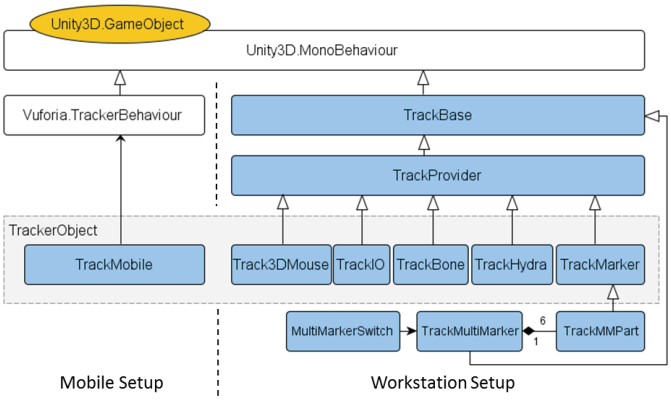

Fig. 4. Tracking class hierarchy

All concrete tracking subclasses provide a consistent tracking data layer, which can be used as tracker object for further processing within the interaction framework.

2) Interaction Framework

Raw tracking data which is fed into the transformation node of a virtual scene *Unity3D.GameObject* should be controllable to provide interaction techniques (IT) for 3D objects selection and manipulation. Therefore, we developed an interaction framework, which is illustrated in Fig. 5.

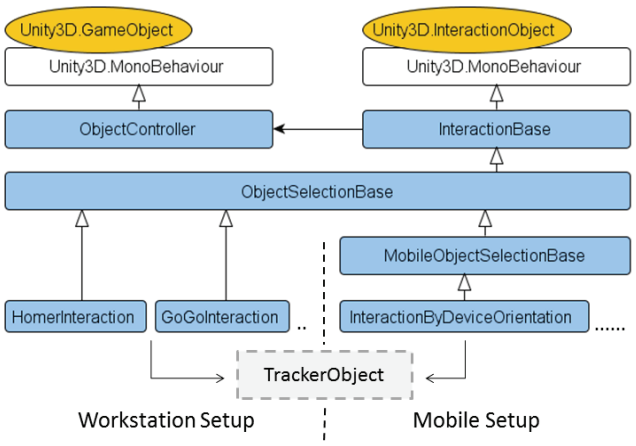

Fig. 5. Interaction class hierarchy

A tracker object is used to access raw tracking data. This data is then processed by the specific IT and handed to the interaction framework. The abstraction layer *ObjectSelectionBase* provides a straight forward and clean interface of data handling for workstation as well as mobile setups and offers a transparent layer to integrate new techniques into the framework. The only information which must be handed over to *ObjectSelectionBase* is a list of the selected object(s) and the absolute pose of the interaction object, calculated by the IT. This data is then processed by the *InteractionBase* class and delivered to all selected virtual scene objects. Virtual scene objects which should be selectable must have the *ObjectController* class attached. Depending on the given pose the *ObjectController* manipulates the position and orientation of the selected scene object.

As concrete 3D interaction techniques, we implemented several standard VR interaction metaphors such as a simple VirtualHand, GoGo [14], Aperture [4] and HOMER [2]. For interacting in a three-dimensional manner on the mobile phone, we adapted the HOMER interaction technique by using the physical pose of the device for object manipulation.

3) Collaboration & Distribution Framework

To provide multi-user support for interaction with different interaction devices and collaboration on one scene over large distances, we implemented a collaboration and distribution framework. It is loosely coupled with the interaction framework and enables distribution for mobile as well as for all workstation setups. The network functions are based on the Unity3D network layer using UDP for communication. We are using a client-server architecture with a direct connection between the server and all clients (star topology). For data exchange remote procedure calls (RPC) and state synchronization are employed. To prevent data loss the state synchronization is buffered.

An overview of the distribution framework and its connection to the interaction framework is given in Fig. 6. The *NetworkBase* class provides functions to initialize the server and to connect a client to the server. All connected clients are managed by the *UserManager* class, implemented as singleton.

To reduce necessary hardware for realizing a client-server application and to improve overall usability, one device can act as server and client simultaneously.

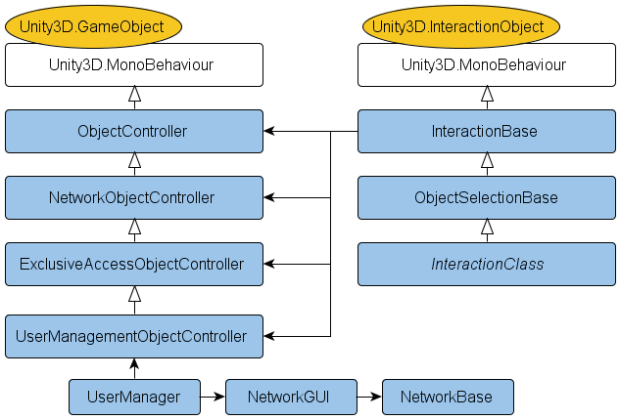

Fig. 6. Distribution class hierarchy

For distributed collaboration each selectable scene object must attach a *NetworkObjectController,* which distributes selection and manipulation functionality over the network. To enable exclusive access of a scene object *ExclusiveAccessObjectController* prevents simultaneous usage by multiple users. As long as a user manipulates the scene object it is locked for other users. To provide exclusive object access to a specific user the *UserManagmentObjectController* is used.

4.4 VR/AR Application Prototyping

In the following we briefly describe the workflow to build a new VR/AR application. First, a new Unity3D project is created and the ARTiFICe framework is added to the project folder. If using a mobile device, the Vuforia Unity3D extension must be included into the Unity3D project as well. All workstation based input devices are configured in the single OT configuration file. Cameras, lights and scene objects are then added to the virtual environment using the Unity3D graphical scene management. They are encapsulated as *Unity3D.GameObjects*. Virtual entities which act as tracker-, interaction- or selectable scene objects are subsequently connected to the according classes of the ARTiFICe framework. Finally, the project is built and deployed to the desired platform and run as single or multi user VR/AR application.

4.5 Supported Devices

Currently, ARTiFICe supports the use of multiple ARToolkit+ markers, the 3D Connexion SpaceNavigator, Microsoft Kinect, Razer Hydra, 6DOF devices tracked by iotracker and mobile phones using Android 2.1 or higher. In addition, all devices that are supported by OpenTracker and VRPN can be used in ARTiFICe.

V. RESULTS

Our framework was intensely tested and evaluated using different setups (section 5.1). Currently, the whole framework runs on Windows and most parts of it (except Kinect and ARToolkit+) on Mac OS X, too. In addition, ARTiFICe was utilized to develop AR applications for the Android platform.

ARTiFICe was used for the master's degree course "Virtual Reality Lab Exercise" at Vienna University of Technology during winter term 2011/12. 80 students built four small VR/AR applications using ARToolkit+ markers for interaction. Subsequently, students developed a distributed and collaborative VR/AR application using various interaction techniques with 3D Connexion SpaceNavigator and Microsoft Kinect. Additionally, our framework was employed for the lab exercise "Augmented Reality" as a part of the master's degree program "Mobile Computing" at the University of Applied Sciences Upper Austria in winter term 2011/12. 17 students were able to develop a mobile distributed and collaborative VR/AR application within just four weeks. Besides that our framework is currently used within a number of on-going research projects which involve fully immersive setups, distributed mobile interaction, 6DOF desktop interaction using Razer Hydra and the Sony Move controller.

By focusing on a well-defined virtual scene management, loose coupling of input devices and interaction techniques, we created an environment which allows technically experienced users to adapt the framework to their needs for application development. In addition, we provide an easy to use framework to help students getting over the initial hurdles of creating quick prototypes of an embodied AR experience.

5.1 Setup Examples

In the following section, we present different hardware setups and interaction devices in various applications, all based on ARTiFICe.

1) Mobile Setup

In Fig. 7, a collaborative and distributed mobile application is shown providing an interactive AR game with physics elements. Therefore, an arbitrary image is tracked with a mobile phone and a virtual scene is mapped onto this image within the mobile display. Multiple users can collaborate, either by pointing their phone on the same physical image or at different physical images showing the same motive. The user on the left hand side currently manipulates a virtual block by HOMER interaction technique while the user on the right observes this interaction.

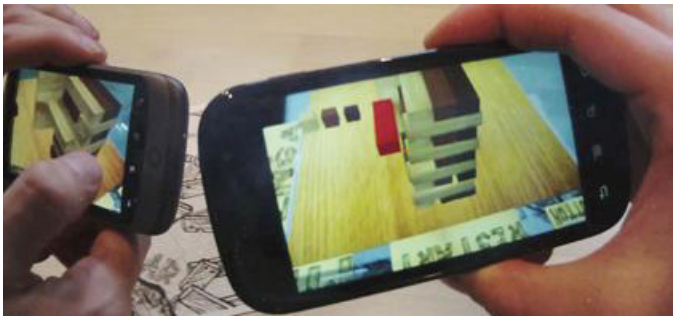

Fig. 7. Collaborative & distributed mobile AR

2) Desktop Setups

A VR desktop application was realized using Razer Hydra as a highly accurate 6DOF interaction device. In an application for geometry education virtual scene objects are controlled using the Hydra, as illustrated in Fig. 8. A collaborative and distributed desktop AR application using multiple ARToolkit+

markers forming a MagicBook [1] is shown in Fig. 9. An ARToolkit+ cube is used as multiple-purpose interaction device.

Fig. 8. Hydra Setup

Fig. 9. ARToolkit+

3) Desktop and Semi-Immersive Setup

In addition to pure VR/AR desktop setups, we used ARTiFICe to realize combined desktop and semi-immersive environments. In Fig. 10 and Fig. 11, a collaborative and distributed multi-user scenario is shown. User1 controls the speed and direction of a flying object by gesture recognition and motion capture using Microsoft Kinect (Fig. 10). User2 interacts with a 3D mouse controlling the height of the flight and clearing the object's flight path using GoGo interaction technique (Fig. 11).

Fig. 10. User1 interacting by motion capture

Fig. 11. User2 interacting with a 3D mouse

User2 uses a 2D desktop visualization whereas user1 is interacting in front of the depth imaging device seeing a 3D visualization provided by a stereo projection with shutter glasses.

4) Immersive Setup

We used our framework to develop a server-client application to provide training for prosthesis patients. The software consists of a server application to control all parameters and a client application to visualize the virtual environment in the HMD. In Fig. 12, a demo setup of this fully immersive application is shown; a 6DOF rigid body target is used to track the user's arm for controlling the virtual prosthesis. Tracking of HMD- and arm target is done using iotracker.

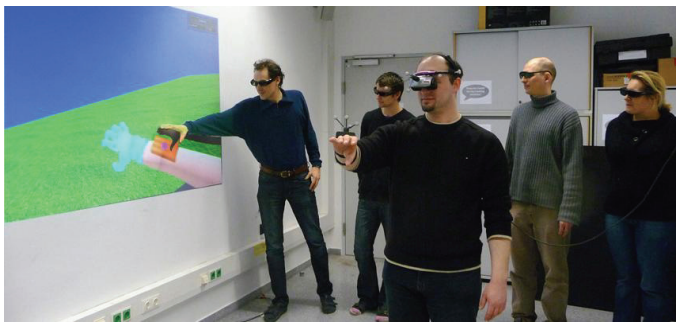

Fig. 12. Immersive setup with 6DOF passive targets

Stereo projection provides the HMD view to an audience to share the HMD user's experience for discussion and explanations.

VI. Future Work

Currently we are working on a proper Sony Move integration in ARTiFICe by implementing OpenTracker support and blob tracking for sensor fusion. More details can be found on the open source Sony PlayStation Move project site [11].

We will focus on improving mobile support and interaction. Therefore, we are currently evaluating the concurrent usage of a semi and fully immersive setup in combination with a mobile phone. Here, we aim on the flexible management of provided user interaction depending on the used device. Furthermore, integration of other mobile AR frameworks besides Vuforia will be assessed and testing of the framework on iOS is planned.

Moreover, we plan to provide our framework as open source project to the developers and research community.

Acknowledgements

We especially thank our former colleague Mathis Csisinko for his extensive contribution to developing the first prototype. Furthermore, we thank our master students Michael Bressler, David Zeller, Manuel Ivancsics and Clemens Gatterer for their additions to the framework by iotracker and Hydra integration.

References

[1] M. Billinghurst, M., H. Kato, H. and I. Poupyrev, The MagicBook: a transitional AR interface, *Computers & Graphics*, vol.25:5, pp. 745-753, Elsevier, 2001

[2] D.A. Bowman and L.F. Hodges, An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments, *Proceedings of the 1997 ACM symposium on Interactive 3D graphics*, pp. 35ff, 1997

[3] C. Cruz-Neira, A. Bierbaum, P. Hartling, C. Just, and K. Meinert, VR Juggler – An Open Source Platform for Virtual Reality Applications, *40th AIAA Aerospace Sciences Meeting and Exhibit Reno*, USA, 2002

[4] A. Forsberg, K. Herndon and R. Zeleznik, Aperture based selection for immersive virtual environments, *Proceedings of the 9th ACM symposium on user interface software & technology*, pp. 95-96, 1996
[5] G. Hesina, Distributed collaborative augmented reality, *PhD thesis*, Vienna University of Technology, 2001
[6] H. Kato and M. Billinghurst, Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System, *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, USA, pp. 85-94, 1999
[7] R. Kuck, J. Wind, K. Riege, and M. Bogen, Improving the AVANGO VR/AR Framework - Lessons Learned, *5th Workshop Virtuelle und Erweiterte Realität der Fachgruppe VR/AR VDTC,* Magdeburg, Germany, 2008
[8] B. MacIntyre, M. Gandy, S. Dow and J.D. Bolter, A toolkit for rapid design exploration of augmented reality experiences, *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pp. 197—206, 2004
[9] OpenNI "(Version 1.3.2.3)" [Software], Retrieved from: http://openni.org, November 2011
[10] OpenVideo "(Version 1.0.0)" [Software], Retrieved from: http://rpm.icg.tugraz.at/, August 2011
[11] T. Perl, Move API, http://thp.io/2010/psmove/, Visited: Feb 2012
[12] PrimeSense NITE, "(Version 1.4.1.2)", Retrieved from: http://www.primesense.com/, November 2011
[13] T. Pintaric and H. Kaufmann, Affordable Infrared-Optical Pose-Tracking for Virtual and Augmented Reality, *Proceedings of Trends and Issues in Tracking for Virtual Environments Workshop IEEE VR 2007*, USA, pp. 44-51, 2007
[14] I. Poupyrev, M. Billinghurst, S. Weghorst, S. and T. Ichikawa, The go-go interaction technique: non-linear mapping for direct manipulation in VR, *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pp. 79-80, 1996
[15] G. Reitmayr and D. Schmalstieg, An Open Software Architecture for Virtual Reality Interaction, *ACM Symposium on Virtual Reality Software & Technology 2001 (VRST 2001)*, Banff, Canada, 2001
[16] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. S. Szalavári, L. M. Encarnacao, M. Gervautz and W. Purgathofer, The Studierstube augmented reality project, *Presence - Teleoperators and Virtual Environments*, vol. 11, pp. 33-54, Feb 2002
[17] E.A. Suma, B. Lange, S. Rizzo, D. Krum and M. Bolas, Flexible Action and Articulated Skeleton Toolkit (FAAST) "(Version 0.08)" [Software], From: http://projects.ict.usc.edu/mxr/faast/, Nov 2011
[18] R. Taylor, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser, VRPN: A Device-Independent, Network-Transparent VR Peripheral System, *ACM Symposium on Virtual Reality Software & Technology 2001 (VRST 2001)*, Banff, Canada, 2001
[19] Unity3D - Game Development Tool "(Version 3.4.2)" [Software], Retrieved from: http://unity3d.com/, October 2011
[20] Vuforia "(Version 1.0.6)" [Software], Retrieved from: https://developer.qualcomm.com/ , October 2011
[21] D. Wagner and D. Schmalstieg, Artoolkitplus for pose tracking on mobile devices, *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*, pp. 139—146, 2007
[22] 3DVIA Virtools, http://www.virtools.com, Visited: Feb 2012

Annette Mossel is a research assistant and PhD candidate at the Interactive Media Systems group at Vienna University of Technology, Austria. After finishing her master thesis on "Development of an adaptive Level-of-Detail terrain visualization in a client-server based 3D geo-information system" in collaboration with the Fraunhofer Institute for Computer Graphics, Darmstadt, Germany, she received her MS in computer science in January 2007 from University of Applied Sciences in Wiesbaden, Germany.
Her research interests include tracking technologies and applications, Virtual and Augmented Reality with an emphasis on mobile VR/AR and 3D interaction techniques. She was involved in different Austrian funded research projects including psychological research of spatial abilities in AR/VR and long range optical tracking. Recent work also focuses on 3D based interaction in mobile AR as well as tracking and localization technologies of unmanned Ariel vehicles.

Christian Schönauer is a research assistant and PhD candidate at the Interactive Media Systems group at Vienna University of Technology, Austria. After finishing his master thesis on "Skeletal Structure Generation for Optical Motion Capture" he received his MS in computer science in 2008 from Vienna University of Technology.
He was involved in different research projects and corporations (e.g. European Union funded project PLAYMANCER) and since September 2011 participates in a research collaboration with Massachusetts Institute of Technology, focusing on feedback technologies and their applicability in motor learning scenarios. His research interests include augmented and virtual reality, tracking and especially full body Motion Capture. An emphasis is placed on the application of tracking data in Serious Games especially in the medical domain (e.g. "Chronic pain rehabilitation with a serious game using multimodal input," *Virtual Rehabilitation (ICVR), 2011 International Conference on*, vol., no., pp.1-8, 27-29). Recent work is also concerned with tracking and reconstruction from depth images.

Georg Gerstweiler received his MS after finishing his master thesis in media informatics in 2011 on "Development of an Active Motion Capture Suit for Teaching Motion Skills" at the Vienna University of Technology, Austria.
Currently he is a research assistant and PhD candidate at the Interactive Media Systems group at Vienna University of Technology. He was involved in the Austrian funded research project "Real-time Tunnelling Measurement based on Infrared Optical Tracking" and responsible for the calibration algorithm of a stereo camera setup. His research interests include motion tracking, electronic, embedded computing and especially education in VR/AR systems. He is currently working on topics concerning tracking with mobile devices and a VR setup for teaching motor skills.

Hannes Kaufmann is assistant professor at the Institute of Software Technology and Interactive Systems, Vienna University of Technology and head of the Virtual Reality group since 2005. After completing his PhD thesis on "Geometry Education with Augmented Reality" he did post-doc research in the EU-IST R&D project Lab@Future. He managed and participated in various Austrian as well as European Union funded research projects in the fields of virtual and augmented reality, serious gaming, virtual rehabilitation, spatial abilities and educational geometry software. His research interests include virtual and augmented reality, optical tracking technologies and applications, motion capture, mobile applications, display technologies, medical applications of VR/AR, psychological topics (Cyberpsychology), education in mixed reality, 3D user interface design, AR/VR and CAD Integration. He is the main developer of *Construct3D* which is being used in a number of national and international research projects and initiated development of the optical tracking system *iotracker* at the Interactive Media Systems Group.