# ComposAR: An Intuitive Tool for Authoring AR Applications

Hartmut Seichter[*] Julian Looser[†] Mark Billinghurst[‡]

Human Interface Technology Laboratory New Zealand

## Abstract

This paper introduces ComposAR, a tool to allow a wide audience to author AR and MR applications. It is unique in that it supports both visual programming and interpretive scripting, and an immediate mode for runtime testing. ComposAR is written in Python which means the user interface and runtime behavior can be easily customized and third-party modules can be incorporated into the authoring environment. We describe the design philosophy and the resulting user interface, lessons learned and directions for future research.

**Index Terms:** H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities—; H.5.2 [User Interfaces]: Graphical user interfaces (GUI)—

## 1 Introduction

One way to transition Augmented Reality (AR) into the mainstream is to provide authoring tools that allow non-programmers to generate their own AR experiences. This approach is evident in the related domains of Virtual Reality (VR) and 3D gaming. While there are commonalities between AR, VR and computer games, AR is unique in that it connects to the real environment and this link is fundamental to the authoring process.

AR authoring tools can be divided into three levels. At the lowest level are libraries like ARToolKit [7], providing basic computer vision integration. Next, high-level programming environments, simplify the development process by providing infrastructure required to build applications. Finally, GUI-based tools for non-programmers. Our research prototype, ComposAR, fits into this third category.

## 2 Related Work

Over the last decade many tools have emerged to build AR applications. One of the first, ARToolKit [7] provided marker-based registration using computer vision. It spawned many variations but all require the developer to have C/C++ skills and need to link with graphics and utility libraries.

Higher-level programming tools provide additional functionality to develop AR applications. One of the most established is Studierstube [13] a comprehensive AR framework, including scene graph based rendering, networking, tracking support, and content loading. Others include osgART [10] and DWARF [1]. However, these libraries still require an expert programmer.

Efforts have been made to develop AR tools for non-programmers. One approach is to integrate AR into familiar content-creation tools like Blender [2]. The tools in [4] and [6] allow users to arrange content into AR presentations. Lee et al. [9] demonstrated authoring from within AR. However, these tools have focused on how to manipulate 3D models and assembly of AR scenes, but do not support prototyping of interactivity in the AR environment.

[*]e-mail: hartmut.seichter@hitlabnz.org
[†]e-mail: julian.looser@hitlabnz.org
[‡]e-mail: mark.billinghurst@hitlabnz.org

IEEE International Symposium on Mixed and Augmented Reality 2008
15 -18 September, Cambridge, UK
978-1-4244-2859-5/08/$25.00 ©2008 IEEE

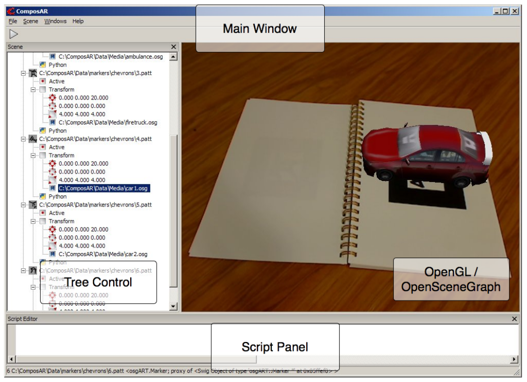

Figure 1: ComposAR interface components

Higher level AR authoring tools address this need for interactivity. DART [11], a plug-in for Adobe Director, inherently has access to the wealth of pre-existing infrastructure. But due to the lack of 3D support DART must take care of this. An approach similar to the one we introduce has been investigated earlier [12] with a more rigid framework. APRIL [8] is an extensible AR authoring platform based on XML descriptions. However, interactions are implemented in non-interpretive languages addressed through the XML parser.

Another approach are visual programming environments such as the ECT graphical programming tool, which was modified to add support for AR input [5]). AMIRE [3], a GUI-based visual AR authoring tool allows describing interactions through visual representations which can become complex.

We propose a pragmatic and extensible authoring tool that supports both scripting and a drag and drop interface, real time interpreted input, and allows users to add functionality depending on their needs.

## 3 Software Design Approach

ComposAR is solely focused on interactive AR, and was designed from the ground up to support AR authoring only. Thus, the main focus of the tool is being able to associate virtual content with real objects and defining interactions for those objects. ComposAR is built atop osgART which uses a plugin architecture to support numerous computer-vision based tracking approaches from which the user can select their preferred method.

An important philosophy is support for the design process, and especially iterative prototyping. Traditionally in AR application development the description of real and virtual is cast in non-interpretive language and therefore hard to change, adapt to new interaction mechanisms, or to transfer from one device to another. In contrast, ComposAR is based on an interpretive language that allows designers to rapidly prototype AR scenes and interactions. In this way designers can see changes instantly and can dynamically change their AR applications until they get the interactivity they desire. Lastly, we have aimed for extensibility; it is impossible for tool builders to anticipate what their tools will ultimately be used for. Therefore to meet the needs of end users, a tool should be able to be extended and customized. ComposAR provides a Python interface allowing users to add their own custom Python modules to support their particular needs.

## 4 System Overview and Implementation

ComposAR is written in Python using various extension libraries. It provides a GUI divided into re-arrangable panels (see Figure 2). The overall design philosophy is to keep ComposAR a pragmatic tool revealing its advanced features only on demand. We intentionally avoid presenting technical aspects such as the projection matrix or the scene-graph branch for the video background. By hiding them we emphasize the AR component of the application, focusing the user's authoring attempts to the marker transformations and their content. These are accessed through a tree layout. A node in the tree structure can be activated with a single click, highlighting the respective 3D scene object and showing manipulation handles. Activating a node facilitates editing, such as file loading, or manual entry of transformation data.

This GUI was implemented in wxPython which is a wrapper for the cross-platform GUI and system development toolkit wxWidgets. In a similar way to wxPython we developed osgPython [1], a comprehensive wrapper for the OpenSceneGraph. Included in this package is the GPL version of osgART [10] with the respective bindings and plugins for ARToolKit and various video input sources. By using these components we can directly foster the wide variety of plugins available for OpenSceneGraph for database loading and writing.

### 4.1 Interaction

What makes ComposAR unique compared to other AR authoring tools is the ability to support different levels of interaction. We follow a similar approach to [5] by extending the notion of a fiducial into a sensor. The intermediate level of the system implements a *Action-Reaction* mechanism imitating Newtons' Physics paradigm. To distinguish the different levels where input and output are connected we describe the chain of events through *Sensors*, *Triggers* and *Actions*.

*Sensors* provide a raw live datastream into the authoring environment. All physical devices including keyboards, mice and other conventional input devices are sensors. The data provided by sensors is elevated to the state of "information" only once it is interpreted by a *Trigger*, which evaluates the input and decides whether or not to invoke an *Action*. An example of this process is the monitoring of the visibility of a marker. Currently ComposAR provides some basic interaction approaches based on a standard repertoire common in AR applications, including interaction based on fiducial proximity, occlusion, tilting and shaking.

Our interaction framework wraps a fiducial as multiple sensors providing streams of typed data. These streams can be connected to a *trigger*. Triggers implement user parameterization to provide a state tied to an *Action*. *Action*s are scripted with two virtual function calls: initialization and update. *Action*s are attached to scene graph nodes. They extend the notion of the node callback common in scene graph programming. With this technique we can not only implement complex interactions, we provide a convenient means of monitoring the behavior of the application.

Another mode, which is important for rapid prototyping of AR applications is the immediate mode. Instead of loading static scripts we enable the user to change code on the fly. In this way the user can interactively write live code and monitor the actual outcome. A typical task for this is to adjust the speed of an animation. Changing the values or calculations live in the editor is directly effecting the display, therefore a simple and powerful means for visual debugging and rapid prototyping.

[1]http://code.google.com/p/osgswig

## 5 Discussion and Future Work

ComposAR is a comprehensive and pragmatic tool for AR authoring adressing users with none or little programming knowledge. In workshops we provided users with a short introduction to 3D modeling and gave them access to ComposAR. Because ComposAR mimics the appearance and functionality of a 3D modeling tool, the connection was easy to understand. The removal of technical AR concepts (i.e. the projection matrix) proved to be quintessential for the adoption by laypersons.

We discovered that, even though there is a low hurdle in learning Python, it would be desirable to create a domain specific language for AR interactions. Various interaction mechanisms require a fair amount of numerical treatment making it difficult to comprehend for a novice. Thus, providing an intermediate level within ComposAR could further enhance accessibility.

ComposAR has proven to be a well rounded base for AR applications in educational, design oriented and research activities.

## References

[1] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riss, C. Sandor, and M. Wagner. Design of a component-based augmented reality framework. In *Proceedings of the International Symposium on Augmented Reality (ISAR)*, Oct. 2001.

[2] P. Grimm. AR blender. http://www.ai.fh-erfurt.de/start/personen/professoren/prof-dr-paul-grimm/projekte/arblender.html.

[3] P. Grimm, M. Haller, V. Paelke, S. Reinhold, C. Reimann, and J. Zauner. Amire - authoring mixed reality. In *IEEE International Augmented Reality Toolkit Workshop*, Darmstadt, Germany, September 2002.

[4] S. Güven and S. Feiner. Authoring 3d hypermedia for wearable augmented and virtual reality. In *ISWC '03: Proceedings of the 7th IEEE International Symposium on Wearable Computers*, page 118, Washington, DC, USA, 2003. IEEE Computer Society.

[5] A. Hampshire, H. Seichter, R. Grasset, and M. Billinghurst. Augmented reality authoring: Generic context from programmer to designer. In *Australasian Computer-Human Interaction Conference OZCHI'06*, 2006.

[6] M. Haringer and H. T. Regenbrecht. A pragmatic approach to augmented reality authoring. In *ISMAR'02: International Symposium on Mixed and Augmented Reality, IEEE*, pages 237–245. IEEE, 2002.

[7] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *the 2nd International Workshop on Augmented Reality (IWAR 99)*, pages 85–94, 1999.

[8] F. Ledermann and D. Schmalstieg. APRIL: A High-level Framework for Creating Augmented Reality Presentations. In *Proceedings of the IEEE Virtual Reality 2005 (VR'05)*, pages 187–194, 2005.

[9] G. A. Lee, C. Nelles, M. Billinghurst, and G. J. Kim. Immersive authoring of tangible augmented reality applications. In *ISMAR '04: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 172–181, Washington, DC, USA, 2004. IEEE Computer Society.

[10] J. Looser, R. Grasset, H. Seichter, and M. Billinghurst. OSGART - A pragmatic approach to MR. In *Proceedings of International Symposium of Mixed and Augmented Reality*, 2006.

[11] B. MacIntyre, M. Gandy, S. Dow, and J. D. Bolter. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. In *Proceedings of the 17th annual ACM symposium on User Interface Software and Technology*, pages 197–206, 2004.

[12] C. Owen, A. Tang, and F. Xiao. Imagetclar: A blended script and compiled code development systems for augmented reality, 2003.

[13] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavari, a. L. Miguel Encarnaç M. Gervautz, and W. Purgathofer. The studierstube augmented reality project. *Presence: Teleoperators and Virtual Environments*, 11(1):33–54, 2002.