Journal of Visual Languages and Computing 20 (2009) 61–79

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Immersive authoring of Tangible Augmented Reality content: A user study

Gun A. Lee [a], Gerard J. Kim [b,*]

[a] Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea
[b] Department of Computer Science and Engineering, College of Information and Communication, Korea University, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 8 August 2007
Received in revised form
25 July 2008
Accepted 30 July 2008

Keywords:
Immersive authoring
Augmented reality
Tangible interface
User study
Interaction design

ABSTRACT

Immersive authoring refers to the style of programming or developing content from within the targetexecutable environment. Immersive authoring is important for fields such as augmented reality (AR) in which interaction usability and user perception of the target content must be checked first hand, *in situ*. In addition, the interaction efficiency and usability of the authoring tools itself is equally important forease of authoring. In this paper, we propose design principles and describe an implementation of animmersive authoring system for AR. More importantly, we present a formal user study demonstrating its benefits and weaknesses. In particular, our results demonstrate that, compared to using the traditional 2D desktop development method, immersive authoring gained significant efficiency in specifying spatial arrangements and behavior tasks, a major component of AR content authoring. However, it was not so successful for abstract tasks such as logical programming. Based on this result, we suggest that a comprehensive AR authoring tool should include such immersive authoring functionality to help, particularly non-technical media artists, create effective contents based on the characteristics of the underlying media and interaction style.

1. Introduction

Augmented reality (AR) is a newly emerging type of digital content that combines real imagery (usually captured by video cameras) with virtual 3D graphic objects. Thus, its content is 3D by nature. Compared to 2D oriented content or applications for which a stable interaction platform exists, developing 3D content (such as AR content) requires a careful consideration of interaction usability and user perception, in addition to the basic functionality. Immersive authoring has been proposed in the virtual reality (VR) community as one way to achieve this objective [1]. Immersive authoring refers to the style of programming or developing content from within the target executable environment. By working directly within the target executable environment, the developer gains a better "sense" (since a full blown formal usability test is not always feasible) for the content in development as seen, used, or felt by the user. Note that the executable environment of AR is quite different from that of the desktop, often requiring the user to wear a head mounted display (HMD), camera, sensors and use non-traditional interfaces, a time consuming process in itself. Thus, immersive authoring has the additional benefit of reducing the period between contents development and testing/deployment.

Immersive authoring is similar to the concept of "What You See Is What You Get (WYSIWYG)," the most prevalent form of visual authoring tool today [2]. While the concept of WYSIWYG is quite intuitive and its benefits have been attested to in theory and practice for some time, immersive authoring is still just an interesting proposal,

[*] Corresponding author. Tel.: +82 2 3290 3196; fax: +82 2 3290 4295.
E-mail address: gjkim@korea.ac.kr (G.J. Kim).

1045-926X/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2008.07.001

without benefits which have been formally demonstrated or verified. This is partly because the efficiency of immersive authoring depends on its own interaction usability and ease of use. Despite potential benefits, any authoring tools, immersive or not, will neither be effective nor gain popularity if they are difficult to use. In addition, one must also consider that some aspect of the immersive content may not be achieved in the most efficient way through immersive interfaces (e.g. specifying logical behavior). Such issues need to be examined in conjunction with each other.

In this paper, we first propose requirements, particularly in terms of its interaction design, for immersive authoring for AR content. Then, we briefly describe our implementation. Our central concept of immersive authoring for AR is an extension of the "WYSIWYG" into "WYXIWYG (What You eXperience Is What You Get)" [3]. We demonstrate the projected benefits of immersive authoring by conducting an interaction usability test with AR authoring tasks, as compared to using the traditional desktop development method.

In the following, we first review some previous research related to our study. We then discuss the design principles and requirements for immersive authoring in terms of interaction usability. We also briefly describe our implementation of an immersive authoring system called iaTAR. Section 4 describes the formal usability test we have performed to evaluate our immersive authoring system. Finally, we conclude the paper with an executive summary of our contribution and future research directions.

2. Related work

The basic idea of immersive authoring has been known for some time, although mainly in the context of VR. Stiles et al. proposed a conceptual VR authoring system called the "Lingua Graphica" in which various elements of the programming language were represented in a concrete manner using 3D objects [1]. Several researchers applied immersive VR (rather than using the conventional desktop CAD systems) to create 3D virtual worlds [4–7]. Similar attempts have been made in the AR area also, for example, to construct virtual scenes from within AR environments. Poupyrev et al. [8] suggested a rapid prototyping tool for modeling virtual aircraft cockpits. The system provided a set of virtual gauges and instruments that can be copied over physical tiles. The users were able to test various layouts using an AR interface. Kato et al. [9] suggested a generic interaction method for manipulating virtual objects within AR environments, and applied it to the task of arranging furniture in a virtual room. Piekarski and Thomas [10] suggested 3D geometry modeling techniques for outdoor AR systems. The modeling system was for constructing virtual representations of physical landmarks while roaming and examining the outdoor scene. All of these works, however, fell short of being true authoring tools, as they did not consider object behaviors.

On the other hand, "immersive" behavioral modeling has not attracted the same degree of attention, perhaps due to the seemingly logical nature of the task. Thus, it is not considered fit for 3D immersive VR/AR platforms. However, many object behaviors can be both logical and spatial at the same time. Although geometric modeling is an important part of an authoring process, in this paper we concentrate on the task of scene configuration, object behavior modeling, and other types of functionality for authoring support. Few others have considered immersive authoring of object behaviors in the manner of Steed et al. [11] and Lee et al. [3]. All these systems explored defining behaviors of the scene and objects within the virtual environment using VR interfaces. For example, in the system by Steed et al., the users were able to view and manipulate the links (i.e. data flow) between virtual objects. This was one of the first systems implemented for immersive behavior modeling. However, the data-flow representation was not general enough to accommodate various types of behaviors that were possible in a typical VR system, and there was arguably no compelling reason or advantage (other than merging the executable and development platform) to employ the 3D interaction or immersive environment to view and interact with the data flow representation. Most importantly, it was difficult to judge the overall comparative benefits of immersive authoring from these works without any formal user studies.

AR content has been developed mostly by programming with specialized toolkits (APIs) [12–14]. As a possible means of relieving the burden of low-level programming, few researchers have proposed the use of abstract mark-up languages and visual tools for specifying AR content [15,16]. With the recent popularity and interest in AR, more comprehensive AR authoring tools have been developed extending this approach [17–21]. These tools typically offer a desktop-based GUI interface with various representation constructs (e.g. data flow, state diagrams, geometry), and an executable window showing the evolving AR content (see Fig. 1). Note that in this situation, a camera, usually fixed, is monitoring the target interaction area. For example, CATOMIR is a desktop AR authoring tool developed under the AMIRE project [19]. Its graphical user interface enables users to create and specify properties of required components and link them to create behavior chains. Users can immediately switch to an executable mode (simply by pressing the compile button) for running and testing the result (simply looking at the AR content window, see Fig. 1). The designers augmented reality toolkit (DART) is also a 2D desktop tool for rapid development and prototyping of AR applications [21]. DART is implemented as an upper layer of Macromedia Director, leveraging its familiar behavior modeling method using scores, sprites and scripts.

While development is still based on a 2D desktop and indirect (not immersive) method, the use of mark-up languages or GUI-based tools does significantly reduce the development time. That is, the AR content being designed can be immediately compiled, executed and displayed using the desktop window. While the view of the content is neither first person nor immersive, the development time is still significantly reduced compared to traditional programming. We posited (and found) that experts