박사 학위 논문

가상 세계의 몰입형 저작 기법

이 건 (李 鍵)

전자컴퓨터공학부(컴퓨터공학 전공)

포항공과대학교 대학원

2009

가상 세계의 몰입형 저작 기법

Immersive Authoring of Virtual Worlds

Immersive Authoring of Virtual Worlds

by

Gun A. Lee
Division of Electrical and Computer Engineering
(Computer Science and Engineering program)
Pohang University of Science and Technology

A thesis submitted to the faculty of Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Division of Electrical and Computer Engineering (Computer Science and Engineering program).

Pohang, Korea
October 29, 2008
Approved by

_____

Seungmoon Choi, Major Advisor

가상 세계의 몰입형 저작 기법

이 건

위 논문은 포항공과대학교 대학원 박사 학위논문으로 학위논문 심사위원회를 통과하였음을 인정합니다.

2008년 10월 29일

학위논문 심사위원회 위원장 최승문 (인)

위원 김정현 (인)

위원 박찬모 (인)

위원 강교철 (인)

위원 한성호 (인)

위원 정순기 (인)

DECE 20023535 이 건, Gun A. Lee, Immersive Authoring of Virtual Worlds, 가상 세계의 몰입형 저작 기법, Division of Electrical and Computer Engineering (Computer Science and Engineering), 2009, 121P, Advisor : Seungmoon Choi, Gerard J. Kim, Text in English.

ABSTRACT

Virtual reality (VR) technology has matured for the past several years and found many practical applications in various fields. As the scale and complexity of virtual worlds have grown, building, testing and maintaining a VR-based application (or virtual world) became much more complex and difficult likewise. However, developing a VR application still mostly rely on programming, and the gap between the development and execution environment has made the situation particularly more difficult for artists and domain experts to create the contents for themselves. The lack of powerful and easy-to-use authoring tool is considered as one of the main obstacles for VR to join in the main streams of digital media. In order to overcome this problem, the author proposes the VR content developers to work within the application environment, i.e., the virtual environment. This model is referred as ***immersive authoring***. By working within the application environment, the engineering of usability and perceptual qualities of the content can occur concurrently during the authoring process. In addition, instead of using difficult programming interfaces, direct manipulation and 3D interactions can be employed for specifying various aspects of the VR-based contents.

In this thesis, the author presents the concept, principles and design approaches of immersive authoring. The virtual world authoring task is analyzed to discover requirements of authoring tools, and the immersive authoring design approaches are suggested for each task. Moreover, the basic structure of immersive authoring system and its common development procedure are suggested in order to provide the foundation of

its implementation. Two prototype immersive authoring systems were developed according to these principles – PiP and iaTAR. By investigating these two prototype systems, the author shows the feasibility of the immersive authoring method.

To validate the postulated advantages of immersive authoring, the author performed a formal usability experiment. The main purpose of the experiment was to show efficiency of the immersive authoring method in comparison to conventional development methods. Under this purpose, the experiment was designed to collect performance data while subjects create immersive contents using both methods. The immersive authoring tool used for the experiment was the iaTAR system, and the counterpart was TARML script editing on a desktop computing environment. The results from the user experiment showed that immersive authoring introduced significant efficiency to the authoring procedure in comparison to conventional development method. Especially on spatial tasks, immersive authoring method showed twice the performance. Although the results of subjective ratings demonstrated no significant difference between two, the number of times subjects referred to the user guide document, and the number of times they got lost during the experiment provided indications of higher usability for immersive authoring. According to the debriefing results, it is confirmed that immersive authoring has strength in specifying spatial features of the content, while conventional method was good for non-spatial authoring tasks.

Although there still exists some problems to be solved, according to the results shown in this thesis, the author is convinced of 'immersive authoring' as a next generation authoring method for VR environments. Being improved by further researches on VR interfaces and virtual world models, the immersive authoring method will undoubtedly provide solid assistance to the VR technology, helping it to grow as a main stream of future media and human computer interface.

Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1. Motivation

As the virtual reality (VR) technology has matured during the past several years, the scales and complexity of virtual worlds have grown steadily as well. Virtual reality has found many practical applications in various fields such as education, training and entertainment, and the contents for these application areas now deserve the same professional approach to productions as for movies, games and multimedia contents. Even though virtual reality has been often touted as the next form of human-computer interaction by tapping into the human's multimodality and the power of content immersion, it still has not caught on the main stream of content development.

One of the primary reasons is that virtual reality remains to be "technical" and difficult for the "artists" to develop interactive contents with. Most virtual reality authoring tools are really programming interfaces [48][52][51] suitable for the programmers. In this context, it is difficult for the artists to work with the programmers because the artists often do not understand the technical details, and these programming interfaces do not serve as an effective mean or language for communication among them. Artists and content producers (in comparison to programmers) like to work with "concrete" virtual objects, through reuse and composition of their forms, functions and behaviors.

In comparison, 2D multimedia contents and game developments are no longer driven by hands of few bright computer programmers. Instead, they are planned, produced, marketed and managed by teams of experts with diverse backgrounds, including artists and designers, under the leadership of a creative director. Thus, what is desired, similarly for virtual reality, is a methodology (i.e., tool or medium) that puts the content artists in the driving seat of the production and a way to easily communicate with the programmers for

realization of their ideas.

Building, testing and maintaining a virtual world (i.e., VR-based application or content) is a very complex process. Developers must integrate disparate bodies of knowledge such as tracking, displays, interaction, computer graphics, simulation, human factors, and on top of it, artistry and content psychology and simultaneously consider many system goals, some of which may be conflicting like functional correctness, performance, realism, presence, and usability, for example. Due to this complex nature, construction of a virtual world often requires many revisions, and changing one aspect of the world will undoubtedly affect other aspects. For instance, different shapes and configurations (positions and orientations in space) can result in different dynamic behaviors. In fact, virtual environment (VE) construction is not only about tackling with the traditional computational and logical errors, but also an exploration task. Developers must find the right combination of various types of constituents of the virtual environment like the objects, display and simulation details, interaction and modalities, etc.

Another important factor in making construction and maintenance of virtual worlds difficult is the gap between the development and execution platforms. Like in any design processes, the development of virtual reality systems usually goes through iterations of specification, implementation and evaluation [14][24]. As one of the purpose of virtual reality is to provide a compelling virtual "experience" in an immersive environment, a correct system and content evaluation must include a first hand (i.e., with first person viewpoint in the environment) validation by the developer as well, and this involves the tedious process of wearing and setting up special devices and displays. This is a major hurdle that creates both temporal and spatial gap between the implementation and evaluation stage, and in turn, causes delay and inefficiency in the development process. Many virtual reality systems have usability problems simply because developers find it costly and tiring to switch back and forth between the development (e.g., desktop) and

execution environments (e.g., immersive setup with HMD, glove, trackers, etc.), and fail to fully test and configure the system for usability.

By definition, virtual reality contents are "immersive" (i.e., user interacts "in" the VE) in nature, unlike traditional PC or console games in which user interacts in third person viewpoint, usually. This makes the production of virtual reality content similar to that of movies in which directors use what is called "mise-én-scene" where the director has to observe, communicate and instruct positions and movements of actors and cameras, up close and perhaps even in the eyes of the actors in the scene [23]. Such an ability to direct the behaviors of actors (or virtual objects) in situ would result in a more correct behavioral specification and provide an intuitive way to add dramatic elements to the content. Many virtual object behaviors, such as motions and interactions, exhibit 3D characteristics that are better designed and tested inside the virtual environment for perceptual (rather than functional) reasons. By the same logic, the perceptual qualities such as the sense of presence (dubbed as the feeling of being in the content, which is one of the main goals of VR systems [32][10][27]) should be improved by working within the execution environment.

In order to overcome these problems in VR-based content development, and to provide efficiency, convenience and easier methods in constructing and testing virtual worlds, the author proposes a concept of immersive authoring.

1.2. Definition

Immersive authoring (IA) is an authoring method for developing VR-based content (i.e., virtual world) that allows the developer to experience and verify the content first hand while creating them through natural and direct interaction within the same environment where the final result is to be used. By definition, in immersive authoring, the development process is held within the same environment where the content is experienced

by the end-users; therefore, developers need not to switch between testing and authoring environments, resulting no gaps between two.

The word ‘immersive’ generally means sensory immersion of a computer display by generating a three-dimensional image which appears to surround the user [30]. Comparatively, within the term immersive authoring, the author refers to immersion into the environment where the final content is being used whether it uses an immersive display or not. Therefore, immersive authoring system not only refers to those with immersive displays, such as head mounted displays, but also with other display configurations for virtual reality (e.g., fish tank VR, mixed reality environments, and others).

The word ‘authoring’ generally refers to a process of creating multimedia contents. While authoring task also implies acquiring images, drawing shapes, editing texts and programming behaviors, the word ‘authoring’ is especially used for the integrated process combining media elements to organize a multimedia content. Authoring usually implies using an interactive software designed for non-programmers. However, there are also some special programming languages designed for authoring task [46], though they are usually prepared for the experts, those who need detailed control, and not for common users. In this research, the author uses the same term ‘authoring’ to refer to a process of creating VR-based contents, especially focusing on the tasks for integrating virtual world elements. The author distinguishes the word from ‘programming’, where the purpose of programming is to create programs (e.g., utility tools, device drivers, system programs, etc.) and not contents, usually.

In this paper, hereafter, the author refers to the user of an immersive authoring system as ‘director’ or ‘content producer’ to distinguish him (or her) from normal target users of the content.

1.3. Research Scope

This research presents the basic concepts of immersive authoring of virtual worlds, possible approaches, design guidelines and task requirements to realize the proposed concept. Prototype immersive authoring systems are implemented based on the constructed theoretical foundation, and the postulated advantages of immersive authoring are verified by conducting user studies with the implemented systems.

Developing a virtual world consist of modeling forms, functions and behaviors of virtual objects within it [14]. Though an authoring of a virtual world also implies modeling geometrical appearances of virtual objects, detailed shape modeling is usually done in a separate task, using 3D modeling CAD tools. For authoring tasks, simple virtual object manipulations, such as translation, orientation for placing them and scaling for transforming the form, are usually sufficient. Therefore, in this research, modeling and designing detailed geometrical shapes of virtual objects are excluded from the subject, and are remained for 3D geometrical modeling researches using immersive displays such as [36] and [18]. Instead of restricting the ability of modeling geometrical shapes, here we concentrate on conducting the overall structure of virtual world content and modeling virtual object behaviors and dynamic features of the virtual world, which much more fits to the purpose of authoring, as mentioned formerly.

1.4. Thesis Structure

Throughout the following chapters, first, we review some previous works related to our study. Next, based on the concept of immersive authoring, design principles and task requirements of immersive authoring systems are established and analyzed to support designing and realizing immersive authoring systems. According to these requirements, possible approaches to achieve the purpose of immersive authoring are explored, followed by a description of the common development process for implementing immersive authoring systems. A couple of implemented systems are illustrated showing how

immersive authoring tools can be realized. In addition, results from user studies are described to provide solid evidences on the postulated advantages of immersive authoring method. Finally, the thesis concludes with summarizing the contribution of this research and suggesting future directions.

Chapter 2. Related Work

Building a virtual world consists of two major tasks. One is modeling the geometry (or form) of virtual objects and composing a scene by placing them in a virtual space, and the other is describing their behaviors and interactions. Virtual world authoring systems, including immersive ones, are expected to provide these two main functions. Since manipulating virtual objects with 3D interaction is common, there have been several research works on modeling geometries and constructing virtual scenes within immersive environments using 3D multimodal interactions [18][36][6][5][20]. On the contrary, little attempts have been made on methods for modeling virtual object behaviors within the virtual environment.

One of the first attempts of carrying out the "programming" task in a virtual environment was put forth in a conceptual virtual reality system called "Lingua Graphica" [33]. The main idea of this system was to represent various elements of the programming language in a concrete manner using 3D objects. However, the concept was never implemented. Steed and Slater developed a system that allowed the visualization of the dataflow among the virtual objects within a virtual space [31]. The users were able to view and manipulate the links (i.e., dataflow) between virtual objects. This was one of the first implemented systems for immersive behavior modeling. However, the dataflow representation was not general enough to accommodate various types of behaviors that were possible in a typical virtual reality system, but more importantly, there was no compelling reason or advantage (other than merging the execution and development platform) to employ the 3D interaction or immersive environment to view and interact with the dataflow representation. Another work, named PiVoT-2 [25], used the State charts [9] to represent virtual object behaviors in a 3D environment. While the users could manipulate the states and the transition links to edit and create behaviors, this system also

did not leverage on the spatial nature and other unique advantages of 3D virtual environments. Still, all of these approaches saw the separation of the execution and development platforms as a big problem in effective authoring.

Within similar context, behavior or function authoring using script language has also been attempted in many tools and products, because scripting enables developers to build virtual worlds easily and rapidly, and immediately reflect results to the execution environment (without compiling). This script-based approach is best exemplified in Alice [41]. However, Alice is designed for authoring interactive 3D graphics, and has limited features for immersive VR application (e.g., 3D multimodal interaction, rigorous performance issues and presence). In the area of entertainment, many 3D gaming engines have also adopted this scripting approach [47][39][43]. Scripts are usually associated with an interactive environment in which it is executed. Such interactive programming saves a lot of time and labor. We can confirm from many visual RAD (Rapid Application Development) tools for 2D WIMP user-interfaces and game prototyping examples.

Similar to the author's intention (for fast evaluation of the virtual world in construction), Holm et al. [11] proposed the concept of "immersive editing." The implementation of his idea ran the desktop scene editor and its simulator for immediate viewing of the incremental and interactive authoring. Under this scheme, an assistant (i.e., client user) could experience the virtual world and give advices to the developer, while the developer modified the virtual world on the desktop editor. Although assistants were also able to apply simple modifications to the virtual scene, the major part of the editing and testing remained to the developer on the desktop environment. In fact, one of their suggestions for future direction was to grant more authoring capabilities to the immersive user.

Authoring virtual reality contents, such as 3D interactive stories, resembles the combined task of directing and scriptwriting a movie or play. The scriptwriter focuses more on the

spoken languages among the actors in order to convey the theme and emotion of the story. The motions and gestures to be exhibited by the actors, and the required scenes/atmosphere are described only in abstract terms. It is rather the job of the director to realize the visual and "3D action" part of the outcome of the script by having an understanding and interpretation of the script in his own unique way. As for virtual reality based contents, there is an added responsibility to worry about multimodality and user directed interaction. While most scriptwriters will simply rely on one's imagination to come up with the abstract and conceptual description of the actions and scenes in the script, directors can work "in situ" and interact with the actors and staff members in many ways (e.g., role playing and demonstration, isolated rehearsals, scene/script changes, etc.), to realize the final production [23]. The work and interaction model proposed is inspired by the job of the director as it pertains much more to the perceptual aspect of how the human user will experience the content.

Chapter 3. Requirements

In this chapter, we investigate the requirements of immersive authoring systems. First, we start with general design principles that immersive authoring systems must meet. These principles provide guidance in high-level designs of immersive authoring systems. As a functional level requirement, the author analyzes the authoring task into several subtasks that are expected to be basic functionalities of an immersive authoring system. Based on this task analysis, we explore the approaches to immersive authoring in the next chapter.

3.1. Design principles

To describe the most basic fundamental design principle of immersive authoring systems, the author coins a new term ***'WYXIWYG'***, which stands for ***'What You eXperience Is What You Get.'*** Like the term 'WYSIWYG (What You See Is What You Get)' [8] in modern graphical user interfaces (GUI), this design principle implies that an immersive authoring system must support instant and intuitive evaluation of the virtual world being built. As the definition of immersive authoring implies, directors must be able to experience the same feeling or experience (not only visual and aural, but also other elements such as tactile or haptic if there are any) how the end-users might feel with the content under development. This provides an instant (or even concurrent) evaluation of the content under construction, helping the directors to understand the current status of the content correctly.

Ideally, any interactive VR content should go through formal tests for checking its usability, level of presence, and other effects for transfer of content. In reality, this is difficult for cost, time and resource problems. The next best thing would be to allow the director to get the feeling of the content as much fast and easy as possible, putting the

director in the user's shoe, and at the same time, ensuring that the director's intention are truly reflected. For this, the author employs a modeless style of interaction, WYXIWYG, in which the director can both develop and evaluate the content without switching to one or the other mode.

The next design principle is to ***utilize direct manipulation*** technique as much as possible. Direct manipulation [8] is another important concept in 2D GUI together with WYSIWYG. This refers to manipulating graphical objects in place and directly. For instance, to move an icon to another position, instead of using indirect methods such as typing commands on the command line interface, 2D GUI users points 'on' the icon and 'directly' drags it to the position where they want to place it. This helps users to manipulate graphical objects easily and efficiently. Similarly, since the immersive authoring environment uses three-dimensional interfaces by its nature, there is no doubt that direct 3D manipulation will provide efficient and intuitive way for manipulating virtual objects. In addition, using direct manipulation technique not only increases efficiency, but also supports keeping the WYXIWYG principle in comparison to other indirect methods that usually need to present additional interfaces and information, disturbing the user in experiencing the content as it is.

Although direct manipulations are efficient for virtual object manipulations in three-dimensional space, they might hide the details of the underlying content models. For instance, it is easy to place a virtual object at a specific position with direct 3D manipulation. However, since direct manipulation deals with the 3D position of an object in an abstract and combined way, it becomes difficult to specify precise numbers to each coordinate value. Regarding this problem, as a third principle, the author advices to provide ***sufficient detailed control*** for the underlying content model. The level of control detail might vary according to the characteristics of the target content type, and the authoring system should be designed to provide an appropriate level of control detail.

Finally, the author suggests ***preserving interface consistency*** between the authoring and the final execution environment. Introducing additional interfaces for authoring tasks is hard to avoid. However, adding different interfaces to the authoring environment implies context switching of the directors' mental activity, and this might distract their attention, delaying the development process. Distracting the director's attentions might not only cause temporal delays to the development process, but also degrade the quality of the authoring virtual world. For instance, the presence, one of the most important quality measures of virtual (or augmented) environments, is degraded by distractions and makes it hard for directors to fully experience the constructed virtual world and correctly evaluate it. Therefore, it is highly recommended to use similar (or at least non-conflicting) interfaces with the target application domain.

3.2. Authoring task analysis

Suppose we would like to construct an interactive VR-based content for a following simple story from the Aesop's fables (adopted from http://www.aesopfables.com).

The Hare and the Tortoise

A Hare one day ridiculed the short feet and slow pace of the Tortoise, who replied, laughing, "Though you are swift as the wind, I will beat you in a race." The Hare, believing her assertion to be simply impossible, assented to the proposal. On the day appointed for the race the two started together. The Tortoise never for a moment stopped, but went on with a slow but steady pace straight to the end of the course. The Hare, lying down by the wayside, fell fast asleep. At last waking up, and moving as fast as he could, he saw the Tortoise had reached the goal, and was comfortably dozing after her fatigue. Slow but steady wins the race.

To realize this story as a VR content, several authoring tasks are required. The objects must be modeled (i.e., geometric shape and configuration) according to the details required by their functions which in turn must be specified as well. For instance, the hare's running perhaps requires modeling of the legs and a rhythmic motion associated with the legs. The initial scenes must be put in place, although they may be subject to later modification. Specific details need to be filled for the object's behavior such as their timing, motion profiles, conditions, triggering events, etc. Noting that the manifestation of the behavior may require a careful selection of multimodal output for the best effect, the director can insert special effects, sound tracks and changing lighting conditions into the behavioral time line. All of these important authoring tasks may be repeated and rehearsed, as the content develops and matures, during which the director will adjust parameters, try different versions, replay and review, and even act out the object's role oneself. The director will also constantly require various types of information to make decisions and carry out these tasks.

As shown in the example above, directors have to carry out various tasks while authoring an immersive content. An authoring system must support these tasks in a way that providing basic functionalities for performing these authoring tasks. In this section, the author analyzes the authoring tasks into several subtasks in order to provide functional requirements of an immersive authoring system. We focus on the major tasks that are necessary for authoring a VR content, while one can think of many other functionalities that can be provided for convenience purpose (e.g., note taking).

Authoring tasks can be categorized into three groups: content construction, content review and version management, as summarized in Table 3-1.

Table 3-1 Authoring task model

Task category	Tasks	
Content construction	Virtual object specification	Form specification
		Function specification
		Behavior specification
	Virtual scene specification	Object placement
		Object hierarchy specification
		Scene wide settings
Content review	Content test runs	
	Information browsing	
Version management	Save and reload the content	
	Composition of different versions	

3.2.1. Content construction

Since the purpose of authoring is to create new contents, content construction is the most important task among other tasks related to authoring. Constructing a virtual world implies two major tasks: specification of virtual objects and composing them into virtual scenes.

Virtual objects needed in the content must be defined and their properties must be adjusted to fit the specification of the content being constructed. A virtual object is consisted of its forms, functions and behaviors as defined in [14]. Form represents the properties of the virtual object including its geometrical appearances, structural information, and other physical attributes. A function of an object refers to computationally elementary operations applied to constituent form (e.g., translate forward, rotate arm, switch geometric model, assign a property value, etc.), while a behavior refers to how individual virtual objects systematically change their forms by carrying out a set of functions over a period of time and under different situations. By specifying these three elements, directors can

author a complete virtual object.

After virtual objects are created, they are not remained separately; they are composed into a virtual scene, placed in appropriate positions with their relationships formed up. Directors need to manipulate virtual objects and arrange them in a virtual scene so that virtual objects would be at their designated positions and poses. While arranging their spatial relationships, directors also need to manage the logical structure between virtual objects, which is usually an object hierarchy represented with a scene graph.

Moreover, other adjustments that affect the whole scene are also needed to be set up, such as global lighting and background sound.

3.2.2. Content review

While authoring a virtual world, the director often will want to review the scene (in action or not), and this requires a capability for the director to navigate through the scene and appreciate it from the first person viewpoint. The navigation can be accomplished in many ways ranging from simple keyboard interaction to more natural gesture based interaction (e.g., walk in place [28], head tracking, etc.). According to the design principle on interface consistency, basically, the method used by the end-user must be provided. By using the same interface for navigation, the director would be able to check up how the users will feel while navigating through the virtual world content.

For convenience during authoring, other additional navigation methods would be useful, such as jumping directly to the viewpoint where the director is currently interested in. However, this might mislead the director to misunderstanding the user experience of the content being developed. Therefore, at least the final review and testing of the content must be held with the same interface that the users are going to experience.

While constructing a virtual world content, the director might need many types of information, including general information of the virtual world (e.g., number of scenes, list of actors and their roles) and statistical system performance data (e.g., frame rate). Some of these are usually provided naturally through the content construction process (e.g., object attribute-value list will be visible when attempting to edit it newly). However, the other ones, such as statistical performance information, might need to be invoked with other interfaces such as a system control menu.

3.2.3. Version management

Another important task the director will often carry out is version management. The most basic function required for this task is saving the constructed content into a file system so that it could be reloaded in the later use. The final content becomes ready for distribution when it is saved into a file system, and the users become able to replay the content by reloading it from the distributed file.

Although the authoring task will end up with the final version of the virtual reality content, during the production, there could be different alternative versions for testing purpose. Content development may go through several takes and rehearsals, and clips of behaviors, spatial configurations, and even individual virtual objects may have to be saved, composited, and stitched together during and after the production.

Authoring tasks other than content construction do not need many additional interfaces to perform within an immersive environment. A simple menu system, available in the virtual environment, would be sufficient to save or load different versions of contents and invoke statistical information during authoring. Moreover, for reviewing the constructed virtual scene, no more than the same setup with the end user interface would be required.

However, content construction tasks, conventionally done with programming, need new interaction methods in order to accomplish them within an immersive environment. Therefore, in the next chapter, the author describes the interaction design approaches to these immersive authoring tasks.

Chapter 4. Approaches

To accomplish the authoring tasks within a virtual environment, there can be various methods to apply. In this chapter, the author summarizes these approaches into three main categories – using virtual desktop interface, metaphorical virtual objects and programming by demonstration technique. The first approach is to make able to use conventional development methods within a virtual environment, while the other two are introducing new authoring methods with three-dimensional interactions. Each method has its own strengths and weaknesses, and they are recommended on different situations, such as different behavior models, different interaction methods and interfaces. Appropriate approaches to each authoring task are suggested at the end of this chapter.

4.1. Virtual desktop interface approach

The most naive approach to immersive authoring is to introduce a virtual desktop interface into the virtual environment, and simply borrow the ready to use authoring interfaces in the desktop computing environment. With a 'virtual desktop interface', the author refers to a virtual object functioning as a modern desktop computer interface. This includes various forms of virtual objects, ranging from a complicated virtual object representing a real desktop computer to a simple virtual widget that mimics a 2D GUI control available in the desktop computing environment.

Virtual desktop interfaces having similar appearances with real desktop computers might have a display, keyboard, mouse and others. With such virtual desktop interfaces, directors may interact in the same way that they use them for programming in the real world. They could be also represented in a simpler way, such as a single flat virtual panel showing program codes in text, or 2D diagrams of visual languages. In this case, directors may interact with them using handwritings, gestures, voice recognition or even

real keyboards, if they are available. Figure 4-1 shows examples of such virtual desktop interface in work. The left image shows using a physical keyboard under a video see-though augmented reality configuration, and the right image shows using it within an immersive virtual reality system [15].

Figure 4-1 Examples of virtual desktop interfaces (right figure adopted from [15])

On the other hand, instead of providing a full-scale virtual desktop interface, simple virtual widgets may be sufficient as virtual desktop interfaces for simple tasks. Commonly used controls in 2D GUI, such as menu, button and slider, could be converted and embedded into the virtual environment, and directors might easily manipulate them using a virtual hand or virtual ray, working just as a cursor in 2D GUI.

Although the virtual desktop interface approach is simple and straightforward to achieve the purpose of immersive authoring, it has some problems to overcome due to the current status of virtual reality technology. Immersive displays, especially head mounted displays, still have not enough resolution for displaying large amount of texts and 2D interfaces, clear enough as we usually see on desktop display devices. In addition, tracking devices are also insufficient for handling precise interactions such as typing keyboards, and usability issues remain with virtual widgets. Though this problem could be overcome as virtual reality technology advances, the virtual desktop interface approach

still has weakness in comparison with other approaches on not utilizing the privileges of 3D visualizations and 3D interactions that virtual environments naturally deserve.

4.2. Metaphorical virtual object approach

Using metaphorical virtual objects to represent computer programs is a classical approach to immersive authoring. Former researches, such as 'Lingua Graphica' [33], dataflow visualization [31] and PVoT2 [25], fall into this category. The main point of this approach is to represent programs or algorithms with 3D metaphorical objects, and specify virtual world properties by manipulating these metaphorical objects.

Similar approaches were made on visual language and visualization field, such as CUBE [21] and VISP [7]. However, they are still not enough to represent practical program codes, such as virtual object behaviors, and some of them are just for visualization, and not for editing or writing new codes.

While directors need to manipulate metaphorical virtual objects with 3D interactions, most of the visual languages, such as dataflow diagrams and state charts, are designed for 2D interfaces. Therefore, they may have limits or inefficiencies to use them within virtual environments, directly. In order to overcome this defect, one needs to develop 3D visual languages that are powerful enough to represent virtual object behaviors, and efficient to manipulate with 3D interactions.

4.3. Programming by demonstration approach

Among the approaches to immersive authoring, the programming by demonstration approach leverages 3D interactions in a most active way. With this approach, to describe a virtual object behavior, directors just need to manipulate the virtual object itself in a 3D space, demonstrating how the virtual object must behave. Programming by demonstration technique was introduced from artificial intelligence field for programming

two dimensional graphical user interfaces [19]. As it is referred as a 'direct manipulation for programming task' [26], programming by typing codes in a text form is replaced by directly manipulating the behaving object itself, showing how they are expected to behave. There were some promising works, such as KIDSIM [29] and Pavlov [38], using this technique for programming 2D animations and games. These works offer strong evidence that programming by demonstration would work well also with virtual environments.

While many cases of programming by demonstration imply inferring rules from user's manipulation, we can also think of naive approaches such as simply recording the motion of an object. Motion capturing is one of the well known techniques in computer animation field, and similar functionalities could be easily provided in an immersive authoring environment, as an application of the programming by demonstration approach.

Although programming by demonstration method has strength on representing spatial motions, non-spatial behaviors, such as increments of energy or weight, are hard to represent with. Gauge-like widgets would help to deal with these kinds of behaviors, allowing users to demonstrate how virtual world should behave. Though these kinds of virtual objects might be thought of as metaphorical virtual objects, using these objects in programming by demonstration approach is distinguished from the metaphorical virtual object approach, in which metaphorical virtual objects represent the structure of behavior itself, while here we demonstrate the behavior by manipulating the virtual objects.

4.4. Applying to content construction tasks

All three approaches described above have their own strengths and weaknesses. For behaviors that mostly consists of three-dimensional motions would get advantages from programming by demonstration, and other authoring tasks that need to deal with conceptual properties could use metaphorical virtual objects and/or virtual desktop interfaces. Different tasks have different requirements and features, and therefore

different approaches may be appropriate to meet their needs.

While looking through their strengths and weaknesses, the author also proposes using multiple approaches and making them to assist each other. Although using multiple approaches could leverage the benefits of each method and help to overcome the weaknesses of using only a single method, developers of an authoring system must be careful when they try to combine multiple approaches so that they would not suffer from the increment of the system complexity and the cost for developing the authoring system.

In the rest of this section, the author reviews through the authoring tasks, analyzed in the previous chapter, suggesting appropriate approaches to each subtask. The author especially concentrates on the content construction tasks, since interaction methods needed for performing the other authoring tasks (i.e., content reviewing and version management) within an immersive environment are pretty straight forward and have little issues with designing a new interface, as mentioned earlier in the previous chapter.

According to our task analysis, content construction involves two main tasks: specifying virtual objects and constituting virtual scenes. Directors need to create individual virtual objects and arrange them into a virtual scene, specifying their spatial and logical relation ships.

4.4.1. Virtual object specification

A virtual object is consisted of its forms, functions and behaviors [14], and directors can create a virtual object by specifying these three elements.

Specifying forms of a virtual object can be thought of as assigning values to the data structure that represents the properties of the virtual object. Various data structures with various data types are needed for representing virtual object properties. Scaling value of

a virtual object can be represented with a single numerical value, while three-dimensional positions need vectors, and other composite transformations require matrices. Moreover, some properties, such as the physical appearance (or shape) of a virtual object requires more complex data structures, like polygonal meshes.

Efficient interaction methods vary for different data types and data structures. There is no doubt that 3D manipulations are efficient for changing positions and orientations of a virtual object in three-dimensional space. However, other data structures, those are not related with spatial features, might work better with 2D (e.g., mouse) or 1D interface (e.g., push buttons). Therefore, for specifying those properties involved with spatial features, one could suggest that the programming by demonstration would be the appropriate approach, while for specifying other properties, the metaphorical virtual object or even the virtual desktop interface approach would be the right choice. Table 4-1 summarizes representative data types commonly used for specifying virtual object forms, along with suggested approaches for each data type.

Table 4-1 Common data types used in virtual object forms

Data type	Related properties	Suggested approaches
Boolean	Visibility	VDI, MVO, PBD
Scalar	Scale, Weight	VDI, MVO, PBD
Vector	Position, Orientation	PBD, VDI
Matrix	Transformation	PBD, VDI
String	Name	VDI
Mesh data	Appearance (shapes and textures)	PBD (3D sketch)

* VDI: Virtual desktop interface approach
* MVO: Metaphorical virtual object approach
* PBD: Programming by demonstration approach

Functions of a virtual object are mostly decided by their related forms and their data type. For instance, forms with single numerical values might require arithmetical functions (e.g., assignment, addition, subtraction, multiplication, division, etc.), while those with vectors would require vector operations (e.g., vector addition, vector multiplication, inner product, etc.). For some virtual object properties (or forms), additional functions might be useful, such as a 'rotation' function which can be applied to the property representing the orientation of a virtual object. Therefore, when the basic virtual object properties and their data types are identified through a domain analysis, most of the required functions can be also identified and implemented into the system level (during the development of the authoring system), rather than specifying them in the content level (when authoring a content).

However, if new functions are to be specified, the best and most familiar method is to specify them through typing in textual scripts or code fragments. Directors may need to edit codes for the new function, compile it and reload the module into the authoring system. While defining new functions within an immersive setup would be rare, this can still be realized simply with the virtual desktop interface approach, or with the metaphorical virtual object approach, supported by an appropriate visual language.

Describing behaviors can be thought of as filling out the behavior model of the virtual object. According to the chosen behavior model, directors might use different interaction methods and user interfaces for specifying behaviors. A behavior model is a template of all possible behaviors that virtual objects behave and interact with other virtual objects (including avatars that represent human users). Thus, defining a behavior model entails defining the model for the virtual object and furthermore the virtual world.

Most of the current virtual reality development tools (programming libraries) use conventional general-purpose programming languages for describing virtual object

behaviors, and usually there is no strict model for behaviors. In this case, directors need to edit program codes to describe the behaviors of a virtual object., and this is best carried out through conventional alphanumeric interfaces, i.e., typing with a keyboard. Therefore, the virtual desktop interface would be the appropriate approach to apply, for this case.

Although using general-purpose programming languages for describing virtual object behaviors gives high flexibility, there are also inefficiencies since each behavior must be specified in lots of details causing unnecessary repetition of redundant codes, and it is hard to manage and reuse them since there are no systematic structures. Therefore, some of the virtual reality development platforms provide their proprietary behavior models.

The Virtual Reality Markup Language [50], which is one of the most well known virtual reality platforms, uses the notion of 'routes' to define behaviors. With routes, users can make connections between properties of virtual objects, forming a dataflow graph between them. Alice [41] uses a visual scripting language based on Python language [49] for modeling behaviors, with which users can drag-and-drop each element of the scripting language in a 2D graphical user interface. The user script for describing behaviors is designed in an event-driven form, where a sequence of animation is fired by a designated event, such as a key press or mouse click. Kim et al. [14] uses state charts to describe behaviors. In their work, a virtual object has several states, and the current state is changed according to events. Each state has actions to do when it becomes the current state, when the current state is changed to another state, or during it remains as the current state. Regardless of these works, unfortunately, there is no widely used general behavior model until now.

Reviewing over the behavior models used in virtual reality platforms, we can observe that they can be categorized into two major types: those based on the dataflow model, and the others based on the event-driven model. This also has a connection to the category of

computer simulation systems [1] – continuous and discrete models. When using a dataflow model, the major task for directors is to specify links between virtual world components. For this purpose, immersive authoring systems might provide an interface for manipulating connections between properties of virtual objects, and this implies representing logical entities in a visual form, using the metaphorical virtual object approach. On the other hand, when using event-driven imperative models, directors are to specify events and the following sequence of functions. Although these elements could be also represented with metaphorical virtual objects, since virtual object behaviors frequently exhibit spatial characteristics, three-dimensional direct manipulation and demonstration could be more useful. Directors could constitute a specific situation to represent an event (e.g., collision between virtual objects) by directly manipulating the virtual objects. In addition, specifying a series of actions (i.e., functions with spatial features) could be achieved by simply demonstrating the movement of the virtual object using direct manipulation.

4.4.2. Virtual scene specification

When virtual objects needed in the content are prepared, they must be imported into the scene, placed and oriented at their initial positions. (Note that virtual object behaviors and interactions are usually specified within a virtual scene, therefore, the tasks are not always in a sequential order.) Normally in a conventional development platform, this might be accomplished through a series of trial and error, guessing the translation and rotation offsets of the objects with respect to the parent coordinate system, and then reviewing them when they are visualized. For the immersive case, 3D direct manipulation (e.g., using virtual hands) can be useful where the director simply selects an object and places it on the desired location using one's hand. For objects that are off the operating range, the director can navigate to the vicinity of where the objects must be placed. For an alternative way, the scene can be built in the "god's" point of view as a miniature [34]. This method also takes advantage of direct manipulation, though the

virtual world under operation is shown in different scale and time to time provides third person's viewpoint. In addition to direct manipulation with virtual hands, indirect and constrained manipulations could be useful for finer control. These can be accomplished by introducing virtual widgets, such as rotation/translation axes, or in simpler way, using alphanumeric input devices (e.g., keyboards or props with buttons) to directly type in the precise value.

Besides placing the virtual objects spatially, directors also need to specify the logical hierarchy between virtual objects. Wheels of a virtual car must be not only placed spatially in the right place, but their logical position in the scene graph hierarchy must be also specified in order to behave correctly (e.g., following the body when the car moves). Such operations could be held implicitly by modifying the hierarchical structure based on their spatial relationship. For instance, when a virtual wheel is placed close enough to the car body, the authoring system could implicitly attach the virtual wheel, as a child node, to the car body. However, one would also need explicit manipulations for detail control over the object hierarchy. In this case, the metaphorical virtual object approach and/or the virtual desktop interface approach would be helpful for designing appropriate interfaces for explicit manipulation of the scene graph structure.

In addition to gathering individual virtual objects together, constructing a virtual scene also needs setting up the features that affect the whole scene (e.g., global light settings, fogs, hazes, and background music). These features can be considered as properties of the virtual scene, and can be specified with similar methods that are used for specifying virtual object properties (using virtual desktop interfaces or metaphorical virtual objects). Each supported feature might need to be turned on and off, and some parameter values must be adjusted to meet the requirements of the virtual world.

Table 4-2 summarizes the approaches suggested for each content construction task. While here we only mention the approaches that are thought to be the most appropriate ones, note that other approaches are also worth for consideration under different interfaces or virtual world models.

Table 4-2 Suggested approaches to content construction tasks

Tasks		Suggested approaches
Virtual object specification	Form specification	PBD (3D sketch), VDI, MVO
	Function specification	VDI, MVO
	Behavior specification	PBD, MVO, VDI
Virtual scene specification	Object placement	PBD (3D manipulation), VDI
	Object hierarchy specification	MVO, PBD (implicitly)
	Scene wide settings	VDI, MVO

* VDI: Virtual desktop interface approach
* MVO: Metaphorical virtual object approach
* PBD: Programming by demonstration approach

Chapter 5. Development

According to the established design requirements and the suggested approaches, the author has developed two prototype immersive authoring systems: PiP and iaTAR. In this chapter, the author describes the development details of these authoring tools. Before going into the details of each authoring tool, the author describes the general basic structure of immersive authoring systems that provides a basis for implementation. In addition, the author also arranges common development process for immersive authoring systems according to the proposed general structure. The proposed general structure and development process provide a reference for developing an immersive authoring system, and they can be applied to the cases other than PiP and iaTAR.

5.1. General structure of immersive authoring systems

An immersive authoring system consists of two major parts: the authoring module and the execution module (see Figure 5-1). The execution module runs and displays the content created by the authoring module. A virtual world (the content for virtual reality based systems) are described in a formal model, i.e., a virtual world model, which defines how a virtual world content could be described in a formal representation such as formal languages (e.g., XML). Virtual world models vary according to its target content type. Scene graphs are usually used to describe the structure of virtual worlds, representing virtual objects with nodes and their hierarchical and structural relationships with edges. However, they vary slightly between software platforms (virtual reality programming libraries), and moreover behaviors are usually described separately in another form and this differs from one virtual reality system to another.

Therefore, deciding or defining the virtual world content model (including structural and behavioral models) to use is necessary before implementing the authoring system. After

the virtual world model is determined, the execution module is implemented to interpret and execute the created content according to the standard virtual world content model. In addition, the authoring module is added to work together with the execution module; so that the authoring tasks could be done, during the content is executed. We describe these development steps in detail in the next section.

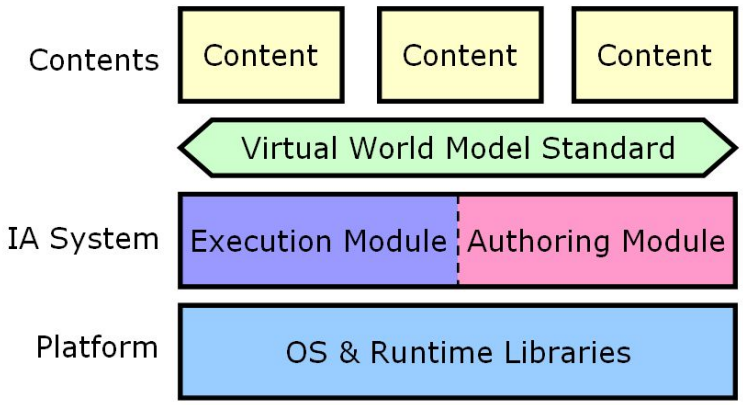

Figure 5-1 General structure of immersive authoring systems

5.2. Common development process of immersive authoring systems

Due to its deep relationship with the virtual world being constructed, the design of an immersive authoring system is tended to be dependent to the adopted virtual world model. Authoring a virtual world can be thought of as filling out the variables or parameters of the virtual world content model. Therefore, according to the virtual world model, the immersive authoring system might use different approaches and this will lead to different interaction methods and user interfaces. In addition to the virtual world model, interaction methods and user interfaces chosen for the target virtual world content also affect the immersive authoring system, because, according to the WYXIWYG principle, they must be provided in the authoring system as well, to be tested with.

Developing an immersive authoring system is neither a simple nor a short-term work. However, since there are common basic structures and similar characteristics, an outline of procedure for developing immersive authoring systems would help to make the process more efficient. In this section, the author outlines the development process of immersive authoring systems.

The process for developing an immersive authoring system consists of four stages: application domain analysis, immersive authoring design, implementation, and verification. The first two stages are for preparing a specification of the immersive authoring system, and the later stages are implementation and verification of the specification established on former stages.

5.2.1. Application domain analysis

The first step of developing an immersive authoring system is analyzing the given specific application domain. The purpose of this stage is to understand the application domain and establish the virtual world model covering the chosen application domain.

First, we can start up with listing up the common requirements of virtual worlds in the target application domain. In this step, common hardware interfaces used in the target domain are set up and the frequently used specific virtual scene structures and behaviors are identified. Identifying the common features of the virtual worlds in target domain helps developers to understand the application domain, and this leads to defining a virtual world model.

Determining a concrete and appropriate virtual world model for the given application domain provides an important foundation of developing an immersive authoring system. A virtual world model would contain a model of required virtual entities (including virtual objects and users), a model for representing the geometrical configuration of a virtual

world (i.e., a scene graph), and a behavior model specifying how virtual objects behave and interact with other virtual objects. Here, virtual objects may include users as avatars, where an avatar can be thought of as a virtual object controlled by the human user through physical interface devices.

5.2.2. Immersive authoring design

According to the virtual world model and the user interfaces decided in the former stage, developers have to decide the immersive authoring method that will fit the requirements best. As described formerly, there are three main approaches to immersive authoring tasks – using virtual desktop interface, metaphorical virtual objects, and programming by demonstration. An appropriate approach (or multiple approaches mixed) must be decided according to the virtual world (especially behavior) model, interfaces and interaction methods used in the target application domain.

As mentioned formerly, especially the behavior model shows a big difference between different application domains, thus gives a big influence on the design of an immersive authoring system. As we referred to authoring a virtual world as filling out the virtual world model, describing a behavior of a virtual object can be thought of as filling out the behavior model, as well. Therefore, the task requirements for describing behaviors differ according to the behavior model. Some behavior models might be enough with an interface for selecting behavior elements from a menu system [41], some others may require more complicated interfaces that can modify connections between given components [50], while others might need exhaustive description of codes (when using a general programming language itself as a formal behavior model). These different behavior models need different interaction methods for manipulating them in an efficient way. Some of them might work more efficiently with 3D interactions (e.g., free motions with motion captures), but others could be inefficient to use three-dimensional manipulation and work better with conventional interfaces, such as keyboards for using

programming languages. Considering these characteristics helps to decide appropriate immersive authoring approach.

After making the decision of what approach to use, visualizations and interactions for the authoring task must be designed. While designing the visualization and interaction methods, developers must beware of the inconsistency between authoring and execution environments. Using the same or at least similar methods with the execution environment may increase the efficiency of authoring system development by preventing developers from writing new codes, and encouraging them to reuse the codes of the execution environment. In addition, it may also help the users of the immersive authoring system (i.e., directors) by preventing them from learning new interactions for the authoring task, lightening their mental loads caused by the context switches between authoring and executing environments. However, although preserving the consistency is highly recommended, developers of the immersive authoring system must remember that some immersive authoring approaches may need certain additional interaction methods and interfaces, and must not disregard them under inordinate adherence to consistency.

5.2.3. Implementation and verification

An immersive authoring system is implemented according to the specification and design accomplished in previous stages. The implementation of an immersive authoring system can be divided into two parts: the execution module and the authoring module.

As described in the previous section, an execution module is implemented to execute a virtual world content represented in a formal virtual world model, which was designed at the application domain analysis stage. The authoring module is built on the execution module, producing virtual world contents in a formal representation (i.e., the virtual world model) by interpreting the director's intentions expressed with interactions.

Since virtual worlds are described in formal representations, they can be easily stored into a file system. Moreover, if the virtual world model is represented in a human readable form (e.g., a scripting language), the development gets much easier. Since the implementation of the execution module can be fully separated from the authoring module, the execution module could be tested independently by loading the contents typed into a script file, instead of relying on the authoring module to produce one.

After implementation, the immersive authoring system needs to be validated whether it fulfills the requirements (the virtual world model specification, interactions designs, and interface usability, etc.) revealed in the designing stages. Although a formal comprehensive user studies are still needed, one simple way to validate the implementation is to perform some case studies. While authoring a number of representative virtual world contents in the target application domain, users can find out improper implementations and unintended problems of the implemented immersive authoring system, and these points can be fed back to the developers and improve the immersive authoring system design. The virtual worlds used for testing the implemented immersive authoring system must have representative features of the target virtual world model. In addition, authoring a number of test virtual worlds, each focusing on some different features of the authoring tool, would help the verification process to be more effective and more comprehensive.

Although the development process of an immersive authoring system is described in a sequential order, the process can be reconfigured under different situations. For instance, in order to develop an immersive authoring system over an existing virtual reality framework, the domain analysis stage can be skipped (or done grossly) and proceed directly into the immersive authoring method analysis step, adopting the pre-built virtual

world model of the framework. Besides omitting some stages, the sequence of the process could also be altered. For example, developers may return to the immersive authoring method analysis step, and modify the system design according to the figured out problems at the implementation or verification stage.

5.3. Prototype systems overview

Two prototype immersive authoring systems were implemented – PiP and iaTAR – where the first one is under a fully immersive virtual reality configuration, and the other one is in Tangible augmented reality (AR) [13] environment. Those two platforms were chosen since they are widely used platforms in virtual and augmented reality researches. Other features of the two systems were also chosen carefully to cover representative behavior models, and immersive authoring approaches. Table 5-1 summarizes the features of each prototype immersive authoring system. The implementation details of each authoring system are described in the following sections.

Table 5-1 Prototype immersive authoring systems

Prototype system	Platform environment	Application domain	Virtual world & behavior model	Main approach
PiP	Immersive virtual reality	Interactive virtual reality adventure	Scene graph / Event-driven	Programming by demonstration
iaTAR	Tangible AR	AR storybook	Component / Dataflow	Metaphorical virtual objects

The PiP system was designed for authoring interactive virtual reality adventure applications. The virtual world model of the PiP system was designed based on scene graph structure, which is common for describing immersive virtual environments, and the behavior model was based on event-driven model, which fits well with interactive

behaviors of virtual objects. Noticing that interactive behaviors of virtual object include many spatial motions, the author chose programming by demonstration approach for immersive authoring design in the PiP system.

With the iaTAR system, the AR storybook, similar to MagicBook [4] which is one of the most representative applications of Tangible AR interface, was chosen as a target application domain. After a careful feature analysis on Tangible AR application domain, a component based virtual world model was chosen in iaTAR, with a behavior model based on dataflow graphs. According to the chosen virtual world model, the main approach to immersive authoring was decided as using metaphorical virtual objects.

5.4. PiP

Here we describe an immersive authoring system for interactive virtual world adventure contents, named 'PiP', which stands for 'Programming virtual world within virtual reality Program'. The PiP system mainly uses the programming by demonstration approach to immersive authoring. Users can place virtual objects in a virtual world and model their behaviors while experiencing the virtual world being built, wearing a head mounted display, data glove, and tracking devices.

5.4.1. Application domain analysis

In an interactive adventure virtual world, participants play a role within a story and interact with the virtual world (i.e., virtual objects) to decide the story line. A story may consist of multiple scenes (or stages), and the scenes are changed according to the participant's decision or the result of simple interactive games. In order to grant these requirements, the virtual world model must support multiple scenes and must be able to switch between them.

Thinking over that scenes of stories are usually similar to the real world environment, and

since most of the participants will be naive users (e.g., children), a virtual hand is chosen for the interaction method. With the virtual hand interface, users can grab virtual objects to select and manipulate, naturally, just as they do with their hands in the real world. For navigating through the virtual world, users can simply point to the direction they want to move with the virtual hand interface. This interaction design naturally leaded to a hardware configuration using a data glove for hand gesture recognition, and tracking devices to track the participant's head and hand movement.

Since virtual objects appearing in the story may behave under its own will and react to the user, the behavior model must support both type of behaviors, i.e., virtual object behaving as time flows and reacting to events such as collisions. A timer event must be also supported, since a story may also change on some specific point of time. The behavior model for the PiP system is designed considering these requirements, while also referencing to the behavior models of previous 3D and 2D interactive applications [41][29][38].

As a result, the virtual world model for the PiP system is designed, considering these features of the application domain. Like conventional virtual reality systems, the PiP system also represents a virtual world using hierarchical scene graph structures. The author extended the usual scene graph structure to include object type (or class) information. This class information, encapsulated in the "type" node as shown in Figure 5-2, serves as a template that the director would fill in to specify features and behaviors of the relevant object. As usual, there is a tree representing the whole virtual environment. The tree has a root node, called "universe," and a collection of "world" nodes representing smaller scenes. Additional data structures are attached to the universe to represent properties of the universe itself, such as the current time, currently activated world, etc. Virtual objects, regarded as instances of the types, are grouped under the corresponding world nodes.

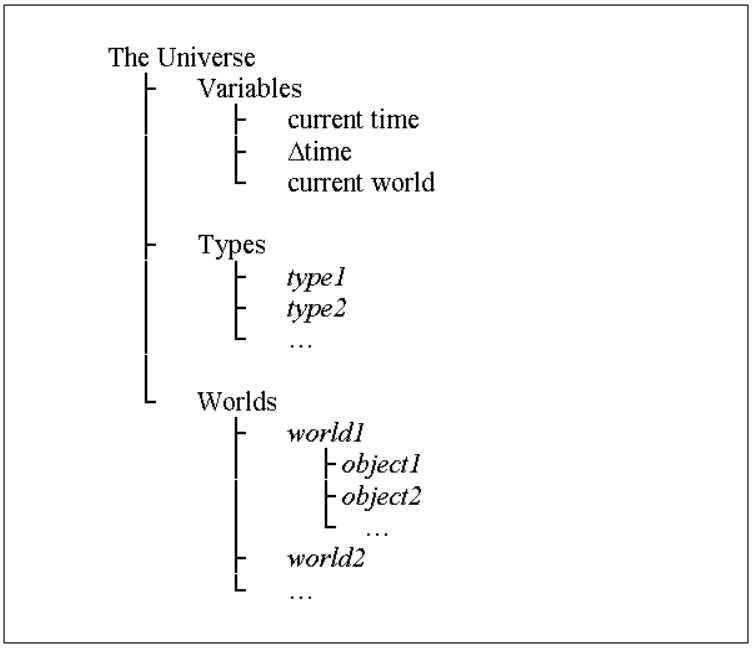

Figure 5-2 The virtual world model in PiP

A virtual object in the PiP system is consisted of its forms, functions and behaviors as defined in [14]. Form represents physical properties of the virtual object including its geometrical appearances and structure. Functions are elementary operations applied to the form of the virtual object (e.g., translate, rotate, scale, create, destroy, and arithmetic operations), and these functions are combined with a set of events to form high-level behaviors (e.g., flee from virtual hand, eat food, etc.)

Virtual objects in the PiP system have system defined form properties and functions as summarized in Table 5-2 and Table 5-3. In addition to these basic properties, users can also add custom properties for defining new types of virtual object, as well.

Table 5-2 Form properties in PiP

Forms	Description
Type	Type of the virtual object
World	The world where the object is located and belongs to
X, Y, Z	3D position of the object
Azimuth, Pitch, Roll	3D pose of the object
Scale	Current scale factors to each main local axis
Appearances	The geometrical model
Sound Clip	The sound of the object

Table 5-3 Functions in PiP

Functions	Description
ChangeVariable	Perform arithmetic operations on form property values. (=, +, -, *, /, %, ^)
Create	Create a new object with given type.
Destroy	Destroy the specified object.
Move	Move the object along the local axis. (Move forward, backward, left, right, up, and down)
Rotate	Rotate the object with respect to the local axis. (Turn left/right, pitch up/down, and roll left/right)
Scale	Scale the object along the local axis.
Play	Play a specific sound clip.

The behavior model in PiP, named ‘ACE (Action while Context holds when an Event occurs)’, is based on general event-driven model, and has three components: event, context and action.

An event is a stimulus that triggers the behavior. There are three predefined types of events: collision, change in object property value, and timer event. Collision event occurs when the object collides with another object. Users can also specify an event that occurs when one of the properties of the object changes its value. Timer events are for synchronizing behaviors by setting them to occur at a specific time, or periodically. Note that a behavior may also be triggered voluntarily without any event. Such behaviors are performed continuously.

A context refers to conditions that must hold when triggering behaviors. There are two types of contexts: spatial and non-spatial. The spatial context specifies the spatial configuration around the virtual object. Regions of interest can be specified around the virtual object with 3D geometries, such as boxes or spheres, and for each region, the director can place a spatial constraint for the object in question with respect to the regions. For instance, in order to specify a spatial context that a certain type of object must present in front of the virtual object in action, one can place a box shaped region in front of the virtual object, and fill this region with another object with certain type that must be there. On the other hand, one can also leave it empty to specify that there must be no objects inside that region. The non-spatial contexts are constraints formulated using properties values of the virtual objects. These can be represented using logical and arithmetic predicates.

An action is a sequence of function calls that are called upon the context satisfaction and/or required event occurrences. Table 5-3 summarizes the set of functions available in the PiP system. Actions are performed sequentially when the designated event occurs and the contexts are satisfied. While most of the actions are performed by the object carrying out the behavior, they can be also applied to the objects that were specified in the event or in the context. For instance, users can specify to perform an action on the collided object or the objects that are within the specific region described by the spatial context.

Figure 5-3 shows the syntax of the ACE behavior model in the BNF (Backus-Naur Form), and an example of its use is illustrated in Figure 5-4. The example describes a behavior of a virtual fish that is triggered when a food type object is collided. Two contexts must hold to perform the sequence of actions: move forward and destroy the collided food object. One is a spatial context that specifies there must be no other objects behind the fish, and the other is a non-spatial context that makes sure the fish is not in a specific appearance.

```
<behavior> -> <actionlist> | <event> <actionlist> | <contextlist> <actionlist>
                  | <event> <contextlist> <actionlist>

<event> ->   collision | status_changed | timer

<contextlist> -> <context> | <context> <contextlist>
<context> -> <spatial> | <non-spatial>
<spatial> -> region_of_interest   type_of_object_in_roi
<non-spatial> -> object attribute relational_op constant
        | object attribute relational_op object attribute

<actionlist> -> <action> | <action> <actionlist>
<action> -> moveforward | movebackward | moveleft | moveright | moveup |
        | movedown | turnleft | turnright | pitchup | pitchdown | rollleft
        | rollright | create | destroy | changevariable
```

Figure 5-3 The structure of PiP script for ACE behavior in BNF

```
<behavior>
        <event collided food>

        <context>
                <roi back    -10 –10 10    10 10 20   empty>
                <thisObject appearance != 1 >
        </context>

        <action>
                <moveforward 1.0>
                <destroy eventedObject>
        </action>
</behavior>
```

Figure 5-4 Sample PiP script specifying the 'eat food' behavior of a virtual fish

All ACE behaviors defined for each object are assumed to run concurrently unless specified otherwise. That is, behaviors can be made sequential or synchronized by specifying the necessary events and contexts.

5.4.2. Immersive authoring design

According to the application domain analysis, virtual hands were chosen as a main interaction method, and this method is used for immersive authoring tasks, as well. Users can manipulate virtual objects and place them in a virtual world using their virtual hands. Figure 5-5 shows translating a virtual fish object with a virtual hand. In addition to direct manipulation with virtual hands, PiP also supports indirect and constrained manipulations for finer control through the virtual widgets as shown in Figure 5-6.

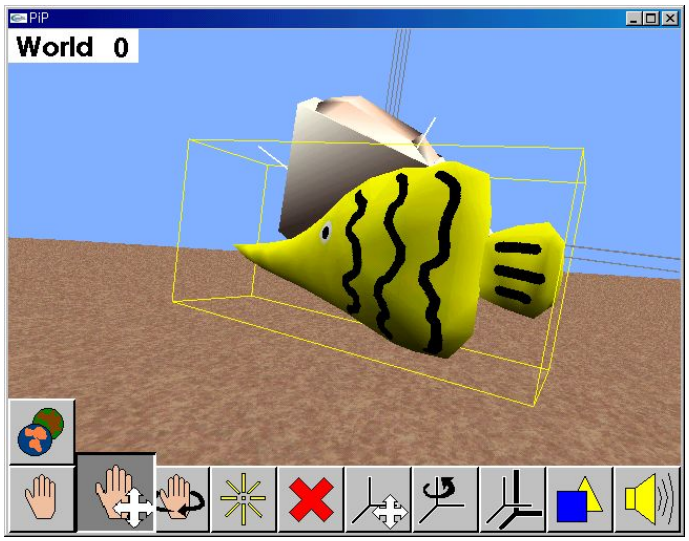

Figure 5-5 Moving a virtual object with virtual hand in PiP

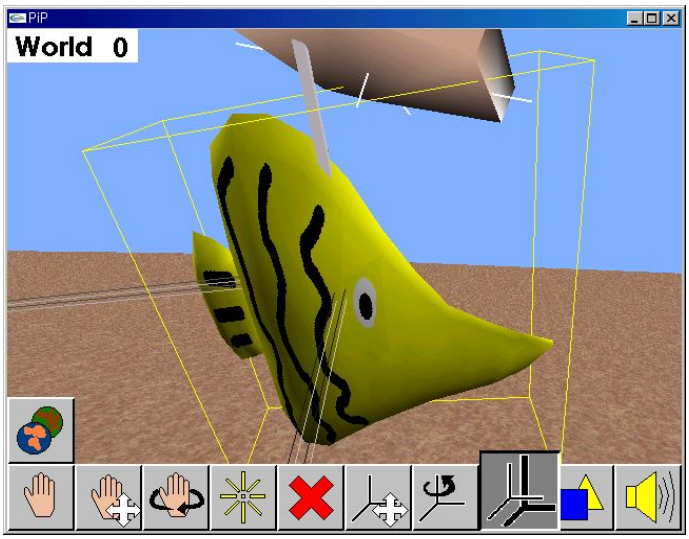

Figure 5-6 Scaling a virtual object with virtual widget in PiP

In contrast with the execution environment, authoring environment needs more interaction techniques and modes for authoring tasks. To meet these needs, a menu system is

necessary to organize these various interaction techniques and authoring tasks. In the PiP system, an iconic menu is presented to the user at a fixed position relative to the viewpoint. As thinking of an interaction method for the menu system, using 3D manipulations with a virtual hand is the most consistent way with the execution environment. However, navigating through many menu items using a virtual hand might bring more fatigue to the participants arm. In addition, continuously moving their hands between a virtual object and the menu system is quite inefficient, while it would be more efficient if menu items were selected in another way and the participant's hand stay with the virtual object being manipulated. Concerning this problem, an additional interface is introduced for the menu selection. Directors hold a 3-button prop on their non-dominant hand, and browse through the menu items and select one by pressing the buttons on it: left, right, and select. By choosing the commands from the menu, directors can change between different manipulation modes (e.g., create new object, translate, rotate, etc.), review the constructed virtual world by playing or rewinding to a specific time, and save or reload the constructed virtual world into and from the file system.

Among the approaches to immersive authoring, programming by demonstration method is chosen as the main manner for describing behaviors in PiP. Modeling object behaviors amounts to using 3D interactions to specify the required events, contexts and actions. More specifically, when authoring a behavior, the director selects the event, context and action modes one after another, and demonstrates each behavior component in each mode to complete the behavior model.

Collision events are demonstrated through manipulating the relevant virtual objects and making them collide with one another (see Figure 5-7). That is, the director can select an object using the virtual hand, and simply move it toward another object to simulate the collision. Changes in object property values can be demonstrated simply by manipulating the virtual object as well. To specify an event that an object is to face toward a certain

orientation, the director would select the object and rotate it toward the appropriate orientation and the final orientation value is stored as the event trigger. Finally, timer events are set through interacting with the virtual timer widget as shown in Figure 5-8.

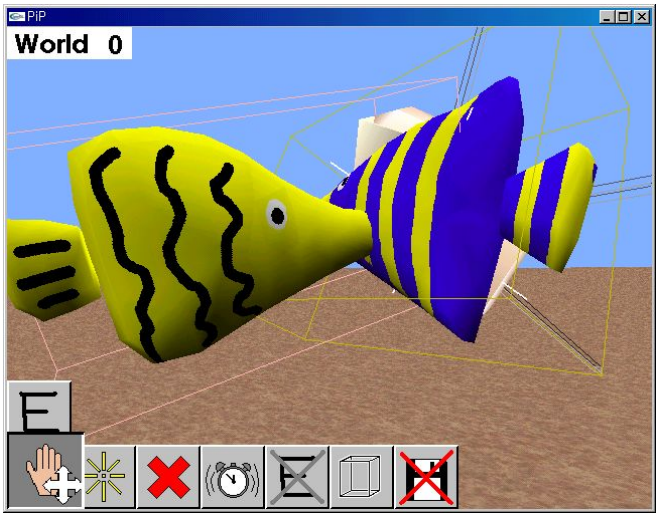

Figure 5-7 Demonstrating a collision event in PiP

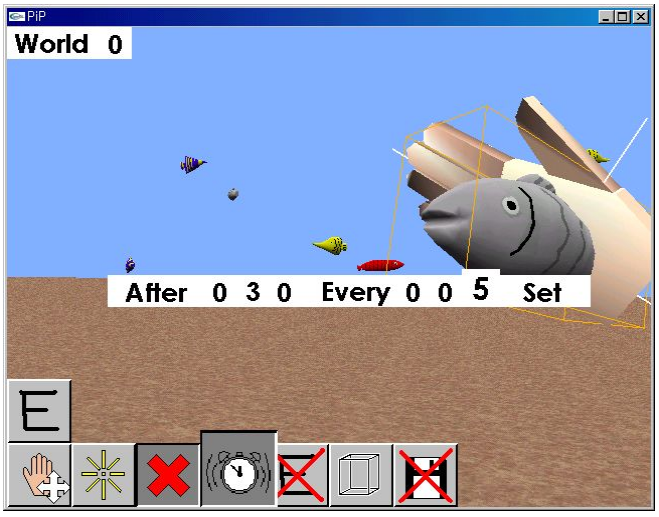

Figure 5-8 Demonstrating a timer event in PiP

Contexts are demonstrated in a similar way. A region of interest in a spatial context is specified using box shaped virtual widgets (see Figure 5-9). After specifying the dimensions of the region of interest box (e.g., using the virtual hand), the director can place a specific type of object inside it, specifying that a certain type of object must present in that region. Setting non-spatial contexts can be specified by simply changing object property values within the context specification step. For example, to specify a condition that an object property must have a certain value, the director simply needs to manipulate and modify the property into a desired value. However, many non-spatial contexts also involve mathematical or logical expressions and applications of various predicate functions. Though simple expressions can be put together using the menu system, it is acknowledged that complex expressions are better to input using the keyboard and mouse.

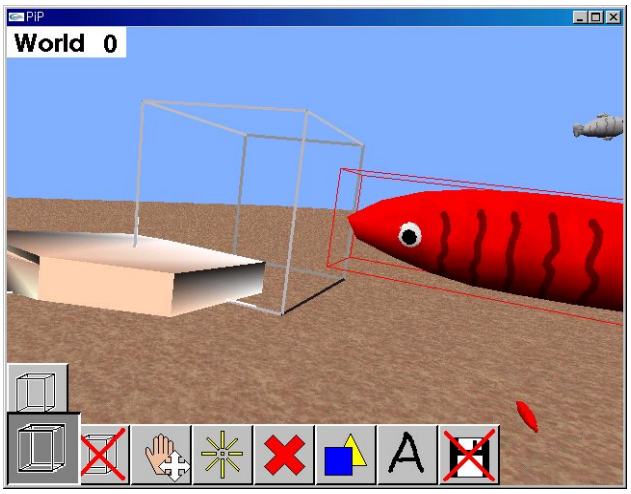

Figure 5-9 Demonstrating a spatial context in PiP

Actions are easily demonstrated by manipulating virtual objects with 3D interactions and menu commands. Specifying spatial actions, such as motion profiles and their timing, can be obtained by grabbing the object in question and moving it in space, while the authoring system records the object's movement. Other actions such as changing

non-spatial property values or creating new objects can be easily demonstrated by choosing corresponding menu items to perform each command.

5.4.3. Implementation

The PiP system is built on the Microsoft's Windows operating system using the OpenGL library. The system runs in two modes: authoring and execution modes. While in the authoring mode, the menu system is shown at the bottom of the (HMD) screen for performing authoring operations. The execution mode simply shows the virtual world in action, and reflects the results from the authoring efforts. Figure 5-10 shows the VR devices used in PiP: a head mounted display, FASTRAK [45] for tracking head and hand positions and orientations, the 5th Glove [40] for hand gesture recognition, and a 3-buttoned prop for menu selection.

Figure 5-10 Devices used in the PiP system

5.5. iaTAR

Another prototype immersive authoring system was implemented for Tangible augmented reality applications – the iaTAR (immersive authoring for Tangible AR).

5.5.1. Application domain analysis

Tangible AR interfaces [13] are those in which each virtual object is registered to a physical object and the user interacts with virtual objects by manipulating the corresponding physical object. As the definition implies, there are mainly two kinds of entities in Tangible AR applications: virtual objects and physical objects.

Virtual objects are the main entities that the users are interested in and want to interact with. They are visualized in various forms, such as 2D images or text and, of course, 3D geometries. On the other hand, physical objects serve as tangible interaction points on which the visualization of virtual objects are overlaid, and where the user inputs are sensed. Tangible AR applications typically use real physical objects as input devices.

Since physical objects mediate the interaction between virtual objects and the user, there should be logical connections between the physical and virtual objects; physical objects that the user physically interacts with, and virtual objects that the user virtually interacts with. For example, in order to draw a virtual object registered on a physical object, the position and orientation data of physical objects are needed to be connected and fed to the corresponding virtual object properties.

The connection between physical and virtual objects can be more than a direct mapping between their property values. For example, suppose that we want to change the size of a virtual object according to the proximity of the physical object to the user's view. The distance should be obtained by calculating the norm of the relative position vector between them, and this requires a couple of arithmetic operations. To represent these logical (or

arithmetic) operations, we introduce another type of entity named 'logic box.' A logic box might represent a single logical (or arithmetic) operator, or even a complex behavior such as controlling joint angles of a virtual character.

Putting all these features together, the author suggests a component based virtual world model for Tangible AR contents. In our model, a Tangible AR content is described with a number of components and connections between their properties. Figure 5-11 shows the overall structure of a Tangible AR content. Using a dataflow model to describe the user interface for virtual environments traces back to an early virtual reality program named Body Electric [12] from VPL. And it also agrees with the previous work of Steed and Slater [31] that investigated on editing logical structures of a virtual world within a virtual environment.

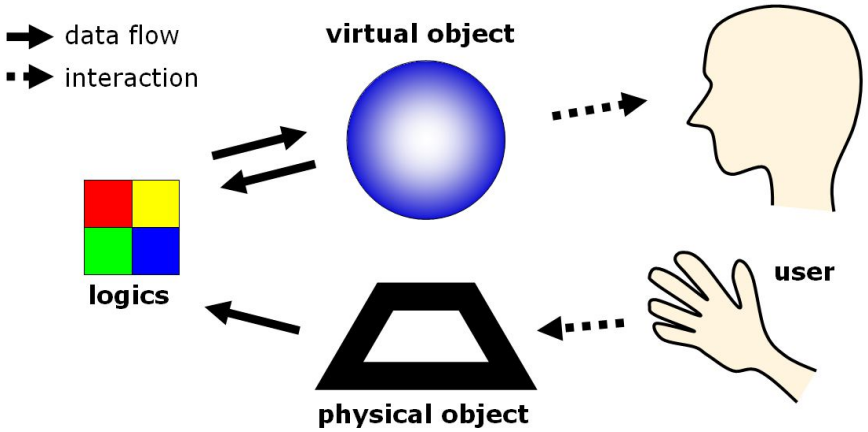

Figure 5-11 Conceptual structure of a Tangible AR content

There are three types of components in our content model: physical object, virtual object and logic box. Each component has a set of properties that represents the state of the component, and each of these properties differs between different component types. Each property has a specific data type and read/write attribute according to the features of the

component it represents. For instance, the position property of a physical object has a vector type value that represents a three-dimensional coordinate. Its value can be read but cannot be modified freely, since it is determined by the physical location of the physical object. Table 5-4 summarizes the properties of each type of components.

Table 5-4 Component properties in iaTAR

Component type	Property name	Data type	Attribute
Physical object	Visible	Boolean	Read only
	Transformation	Matrix	Read only
	Position	Vector	Read only
	Orientation	Vector	Read only
Virtual object	Visible	Boolean	Read and write
	Base transformation	Matrix	Read and write
	Transformation	Matrix	Read and write
	Position	Vector	Read and write
	Orientation	Vector	Read and write
	Scale	Scalar	Read and write
	Play (optional)	Boolean	Read and write
Logic box	(varies)	(varies)	(varies)

Properties of physical objects are mainly related to their tracking results. The visibility of the physical object represents whether the object is successfully tracked, and the transformation, position and orientation properties represent the physical pose of it. Virtual objects have similar properties with physical objects, while they have writable attributes, meaning they can be freely modified. Some virtual objects, that represent sound sources, also have additional Boolean properties for playing and controlling sounds clips. Properties of logic boxes vary from one another. They are determined by the

logical functions that the logic box represents. For instance, a logic box for vector addition might have two input and one output vector properties, while a logic box representing a motor like behavior might have only a single property that gives the rotation value, changing as the time flows.

Tangible AR contents can be authored by connecting a number of components together. Properties of components can be connected to each other when they have compatible data types and attributes. For example, properties with scalar data types can be linked to those with scalar or Boolean values, but cannot be linked to those with vectors, unless they are modified to a scalar value using a logic box. A property used as a target must be writable, while the readable attribute is sufficient for source properties.

Once a property is linked to another, its value is updated according to the source property. For instance, a virtual object can be registered to a physical object simply by connecting the transformation attribute of the virtual object to that of the physical object. For convenience, another property named 'base transformation' was introduced to virtual objects to represent the parent reference frame of the object, and this property is usually used for registering virtual objects onto physical objects.

5.5.2. Immersive authoring design

Given the content model, the task requirements for authoring Tangible AR contents are analyzed. After specifying these requirements, the author describes the interaction design chosen to fulfill the requirements.

As mentioned earlier, the authoring task can be regarded as building application content by describing it with the established content model, i.e., defining entities and filling out their property values declared in the model. Since our content model is a component based one, the main authoring task will be manipulating the components. Directors need to

create components, modify their properties, make connections between them, and destroy it if not needed anymore. Table 5-5 summarizes the main tasks and their subtasks for manipulating components.

Table 5-5 Authoring tasks for component manipulation in iaTAR

Main task	Subtasks
Create	Select type
Destroy	Select a component to destroy
Modify	Select a component to modify Browse & select a property Change the value of the property
Connect (or Link)	Select components to connect Browse & select properties Connect/disconnect the properties

The most basic tasks are creating and destroying the components. For creating a component, the director needs a way to specify the type of the component s/he wants to create. Directors need to browse through a list showing what kind of components they can define. This requires a menu-like interface to a set of items that directors could browse through before choosing one of them. Some vital components, such as pre-defined physical objects, could exist without the need for being explicitly created. These components will be provided to the director from the beginning of the authoring process and the directors would be able to use them immediately. Destroying a component requires the ability to select a component. Directors need to select a component, which they want to destroy, and this requires an interface for pointing or selecting a specific virtual object.

The created components may need to be modified to complete the Tangible AR content. Modifying a component is simply changing its property values. In order to change a property value, the director first needs to select the component and its property that s/he wants to change. This requires an interface for browsing over the list of properties and their values. After the property is selected, directors need to specify a new value for it. The interface for specifying a component property value may vary according to the data type of the property. For example, simple scalar values are easy enough to modify with buttons or keypads while 3D transformations may be more conveniently modified with direct manipulation.

The last main task for manipulating components is to connect their properties with each other. Similar to changing the property values, directors first need to select components and the properties they want to connect or disconnect. Hence, the same interface could be reused for selecting properties, while a new interaction method is needed for specifying the status of their connection.

According to the subtasks identified from the task analysis, summarized in Table 5-5, the interaction methods for each subtask are designed. In order to maintain the consistency between the authoring environment and the final application environment, we avoided introducing new environmental setups. Instead, we only introduced props for the authoring task that can be used in the same environment with general Tangible AR applications. The physical props are simple pads and cubes that are commonly used in Tangible AR applications. Figure 5-12 shows three basic props used for the authoring task: a component browser, a manipulator and a disposer. Since these props are needed only for authoring tasks, and directors can put away them whenever they want, the directors are guaranteed to concurrently experience the final content without any disturbance throughout the authoring task, and this meets the 'WYXIWYG' design guideline.

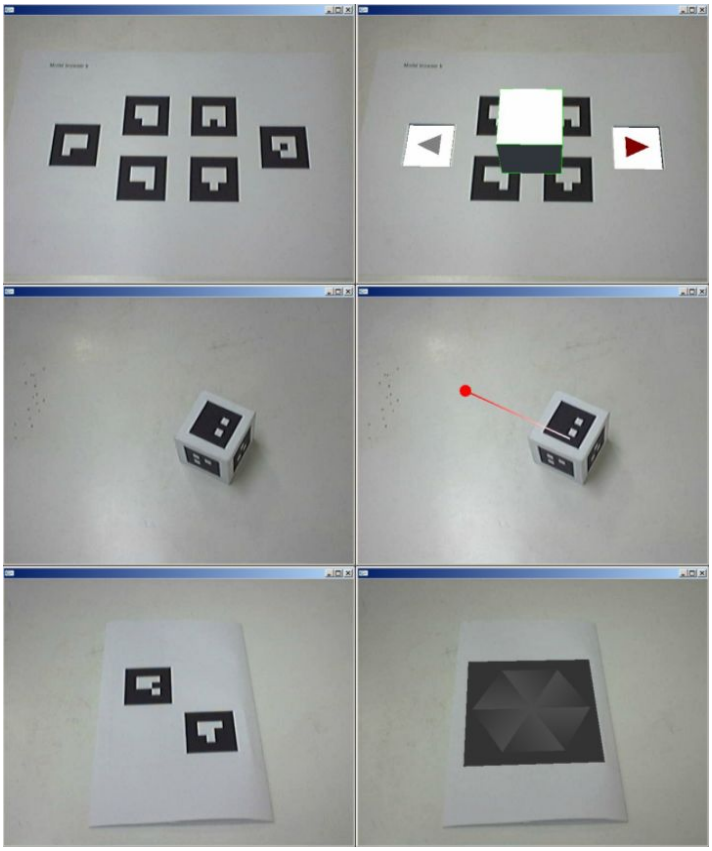

Figure 5-12 Tangible props for authoring tasks in iaTAR

For creating a new virtual object component (or logic box components), directors need to select the type of virtual object they want to create. The component browser provides a physical interface for browsing over available 3D virtual objects and selecting the desired one. Directors can browse over the virtual objects one by one, by pressing (pointing) the arrow buttons on the left and right sides of the browser. To create a new virtual object, directors select the virtual object on the browser by pointing at it with the cube manipulator (shown in Figure 5-13). After a second, an instance of the pointed virtual object is created and selected on the manipulator.

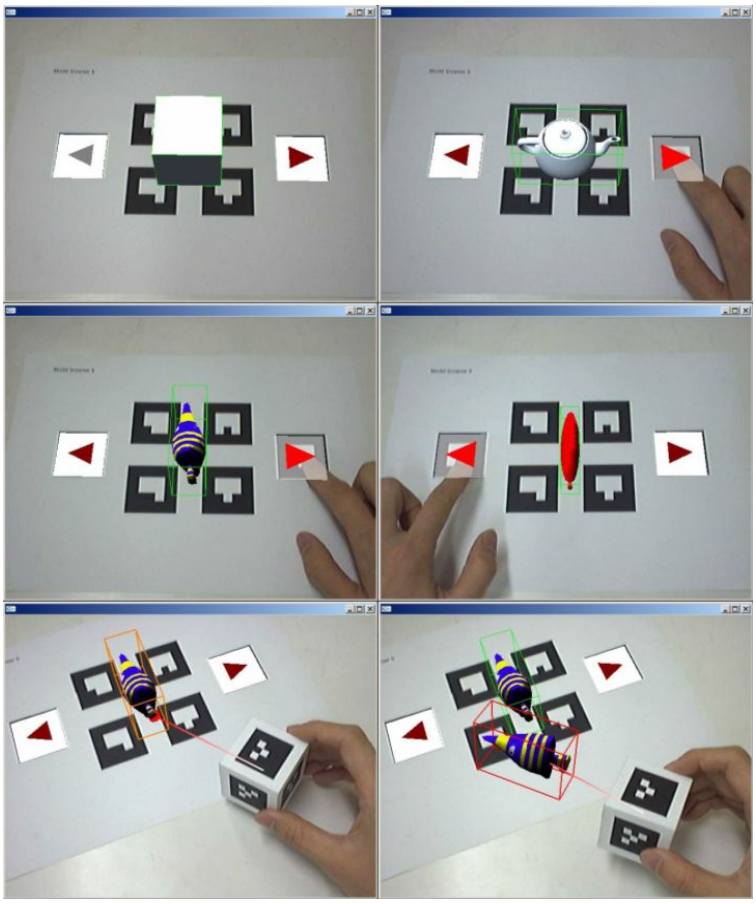

Figure 5-13 Creating a new virtual object in iaTAR

After a virtual object is selected with the manipulator prop, it moves according to the movement of the manipulator. The selected virtual object is kept in a fixed position relative to the manipulator when it is selected, and rotates according to the pose of the manipulator. To release (or unselect) the virtual object, the director simply needs to hide the manipulator for a couple of seconds. The virtual object will remain in the last position and orientation where it was placed. This interaction was designed following the notion of the 'drag and drop' metaphor, which is one of the most well known direct manipulation methods in 2D desktop graphical user interfaces.

The picking up and dropping interaction method is used for destroying objects, as well as for placing (or modifying) them. The upper row of the Figure 5-14 shows moving a virtual object from one physical object to another, while the lower row shows destroying it by dropping on the disposer prop.

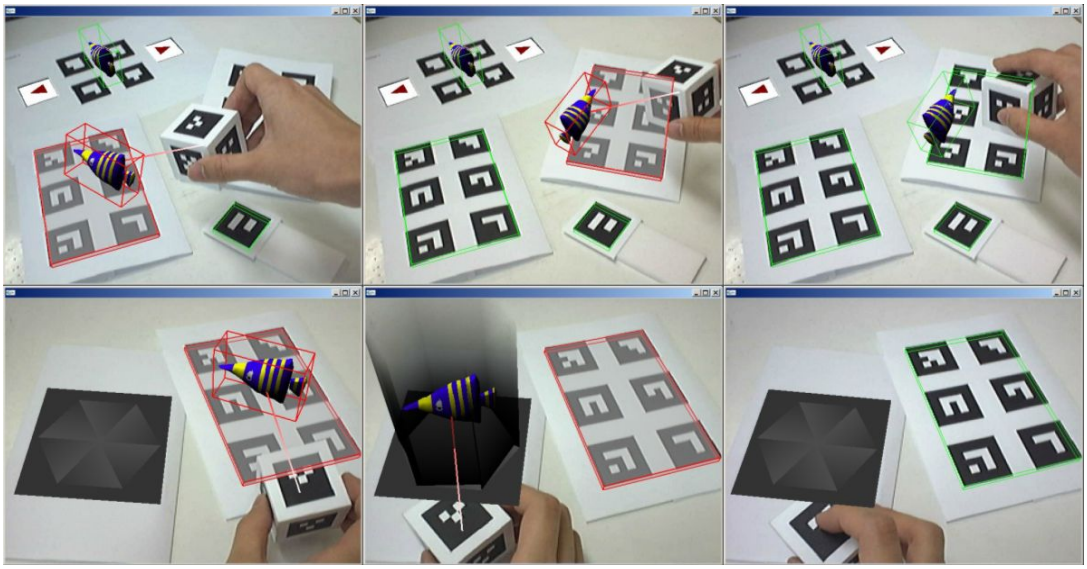

Figure 5-14 Virtual object manipulation in iaTAR: select, move, and destroy

Although direct manipulation of virtual objects takes advantage of the direct 3D manipulation, it hides the details of how the underlying content model is affected. For instance, when placing a virtual object on a physical object, the following things happen. First, the base transformation and visible properties of the virtual object are connected to the corresponding properties of the physical object. Then, the position and orientation properties of the virtual object are changed, in order to place the virtual object in an appropriate position relative to the physical object. In order to provide detailed control over the content under construction, two more types of interfaces, inspector pads and keypads, were added. With these tools, directors can deal with the components in detail

by controlling their individual properties (see Figure 5-15).

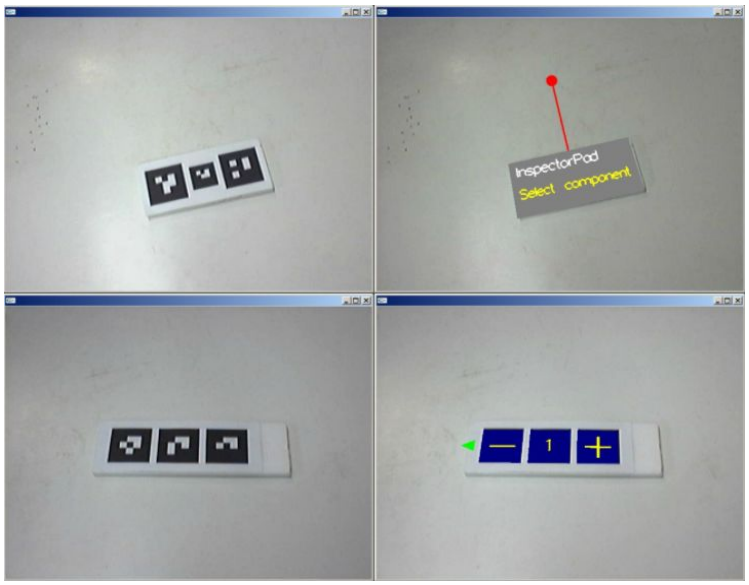

Figure 5-15 Inspector pad and keypad props used in iaTAR

The interaction for selecting and deselecting a component with an inspector pad is similar to that of manipulators: pointing at a component for a second with the probe and hiding the interface. While the manipulators are only allowed to select virtual object components, directors can also select physical objects with the inspector pads.

Once a component is selected, the inspector pad shows the properties and their values of the selected component (see Figure 5-16). Directors can browse through the properties by holding and manipulating the inspector pad. The list of properties shows up when the inspector pad is close enough to the users' view, and the list can be scrolled up and down by tilting the inspector pad up and down. The property displayed on the middle of the inspector pad is the selected one, and the inspector pad shows the value of the selected property. The display format of the value is changed according to the data type of the

selected property, and the small arrows on the left and right side of the inspector pad represent the read/write attributes.

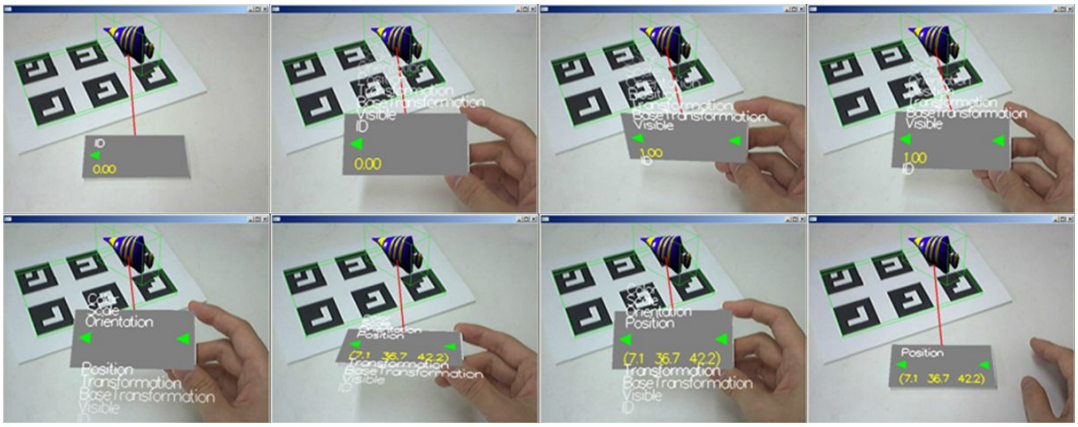

Figure 5-16 Browsing through the property list of a component in iaTAR

To change the value of the selected property, directors can use a keypad together with the inspector pad. Since most of the properties can be represented by numeric values, keypads are used for providing an input method for these. We designed a keypad using occlusion-based interaction method, where a number of visual markers used for tracking the prop are also used for the button pressing interaction (implementation details described in the next subsection). Figure 4 shows an instance of keypad that has '+/-' buttons together with a unit selection button on the middle. Directors can select the unit between 0.1, 1 and 10 by pressing the unit button in the middle, and by pressing the '+/-' buttons, the value is raised or lowered by the selected unit. To change the value of the property selected on the inspector pad, the director can connect the keypad to the inspector pad, and raise or lower the value by operating the keypad. Figure 5-17 shows an example of using an inspector pad and a keypad to change the scaling property of a virtual cube.

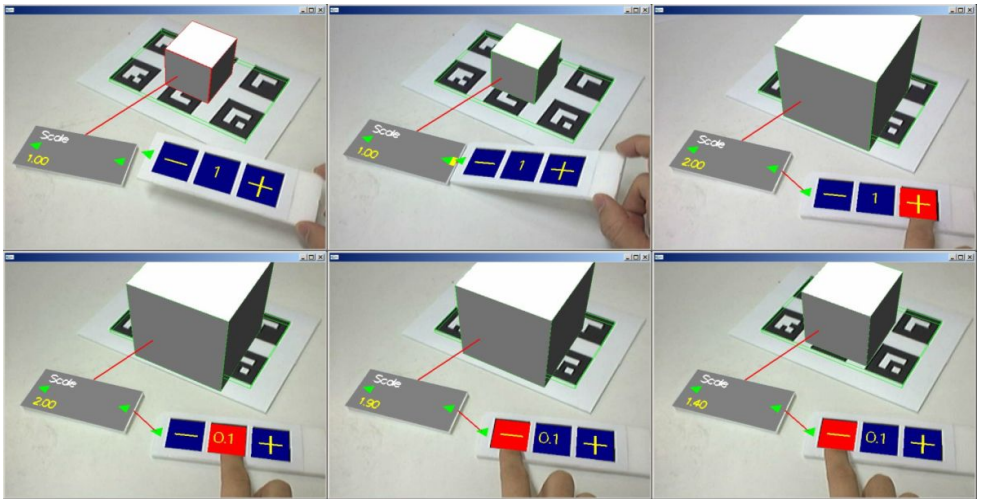

Figure 5-17 Changing the scale property value of a virtual cube in iaTAR

Connecting object properties implies selection of multiple properties. Instead of introducing another selection method, here we simply duplicated the inspector pad to select two object properties being connected.

The interaction method for connecting two selected properties can be designed in various ways. In the early stage of development, first, we have tried to map the logical connection between properties directly to the physical connection between inspector pads, using puzzle cut edges. Although the physical connection worked well as an easy and intuitive input method, since the puzzle pieces were not physically controllable in an automatic manner, it was not feasible for displaying the current connection status unless users connect them physically. Moreover, this early design was poor to prevent incompatible connections (e.g., allowing connections between properties with incompatible data types or attributes). To solve these problems, we have altered the interaction design to toggle between connected and disconnected states when two edges of inspector pads were contacted. Vertical edges of the inspector pads were used as an input and output port of the selected property, and by contacting these edges together, a link was

made (or destroyed) between them if the selected properties were compatible. The same method was used for connecting keypads and inspector pads (see Figure 5-17).

Figure 5-18 shows an example of connecting properties of two components. The visibility property of a virtual fish is connected to the same property of a physical paddle, making the fish disappear when the paddle is hidden by the user's hand.

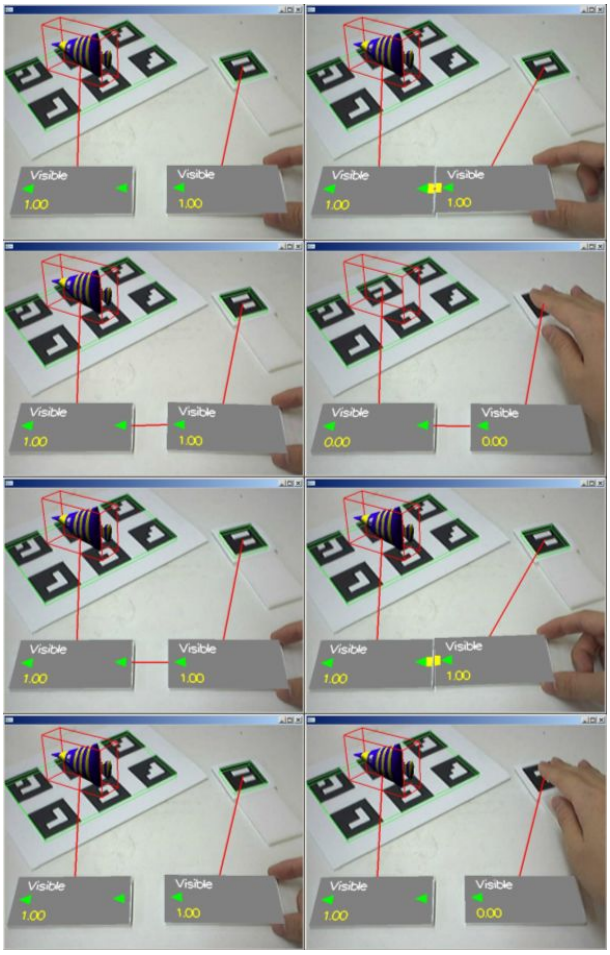

Figure 5-18 Connecting and disconnecting component properties in iaTAR

Simple animations of a virtual object can be specified by connecting virtual object properties to a logic box component representing motions. Figure 5-19 shows an example virtual scene with a windmill augmented on a sheet of paper. The vane of the windmill is animated to spin around by connecting a 'motor behavior' logic box to its orientation property. The 'RPM (revolutions per minute)' property of the logic box was adjusted with the keypad prop to specify the spinning speed.

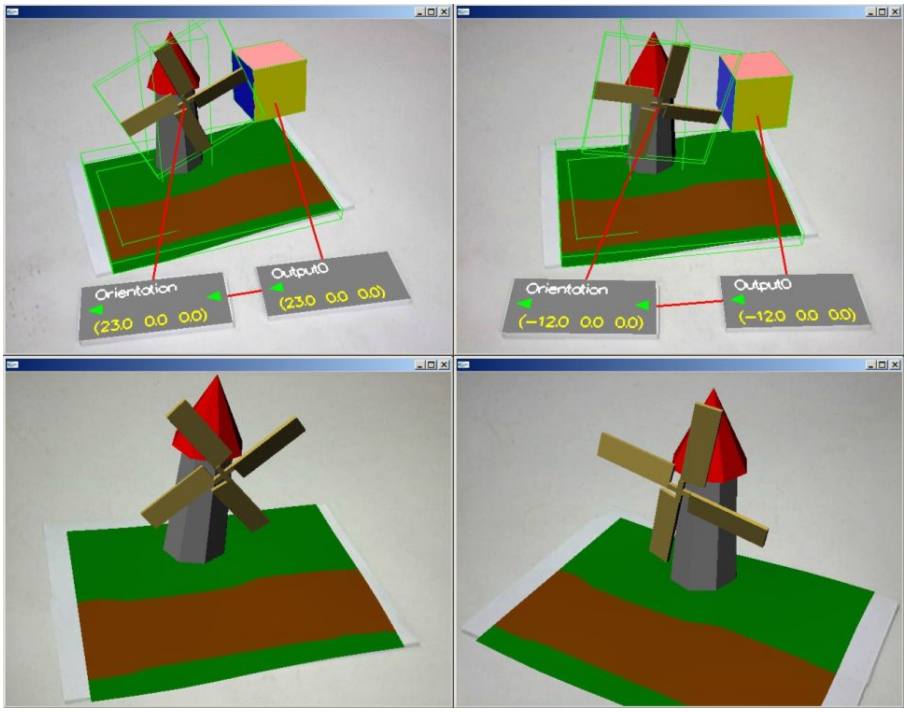

Figure 5-19 Tangible AR scene with a windmill animation

Specifying motions of a virtual object by connecting their properties to logic box components are useful for describing simple regular motions that are repeated with patterns. However, irregular free form motions are hard to describe in this way. Under this notion, iaTAR was designed to support another way to specify virtual object motions, utilizing 3D direct manipulations. The authoring system supports a motion capturing

function, so that the directors could record the motion of a virtual object while they manipulate the virtual object and demonstrate its desired movement. Figure 5-20 shows how a director captures the motion of a virtual fish using the recorder prop. The prop looks similar to a video cassette recorder with record, play and rewind buttons. By operating this prop, directors can control the recording process and play back the recorded motion.

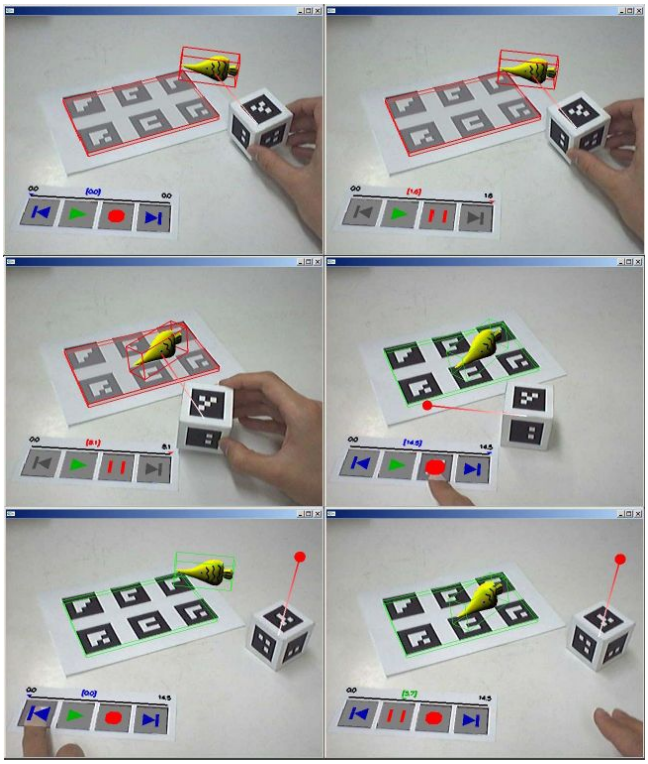

Figure 5-20 Recording a motion of a virtual object in iaTAR

Synchronizing the motions of multiple virtual objects is easily achieved by manipulating multiple virtual objects during the recording process. Directors can use a pair of manipulator props at the same time to record their motions as shown in Figure 5-21. However, manipulating more than two objects is impossible for a single user. Therefore,

directors must be able to synchronize a newly recorded motion with the formerly recorded one. For this, the system is designed to play back formerly recorded motions while recording. Directors can review the formerly recorded motion and start recording at the point where the new motion must begin. In this way, directors can add and combine new motions, synchronized with previously recorded ones.

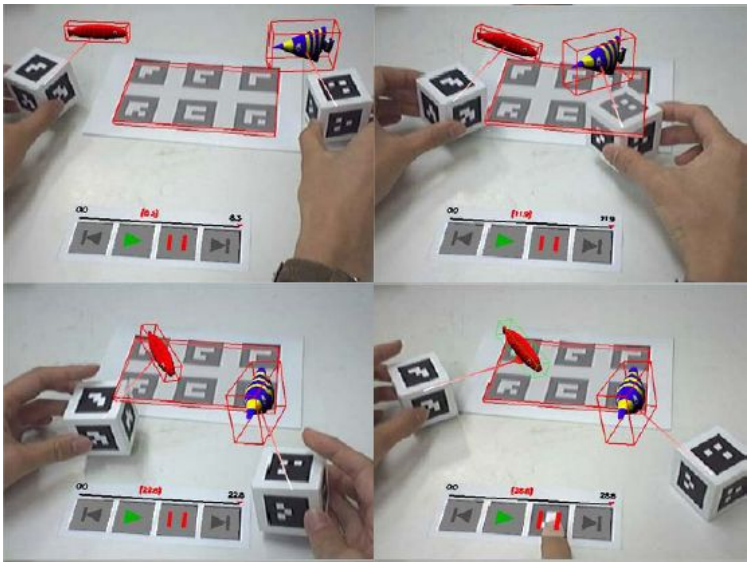

Figure 5-21 Recording synchronized concurrent motions in iaTAR

In addition to providing exocentric viewpoints, AR storybooks (as described in [4]) also provide immersive tour into the virtual world content. In iaTAR, this function is provided by using the navigation bar prop. Users can hold the navigation bar and point on the virtual scene s/he wants to wander within. Changing into the immersive mode, users can see the content as if s/he is within the content and they can move around the immersive virtual environment by pointing to the direction they want to go, using the navigation bar. Moreover, directors can also tweak the virtual environment within the immersive mode using the authoring props, for example, they can reposition a virtual object using the manipulator. After finishing the immersive tour into the content, the user

can switch back out to the augmented reality mode by flipping around the navigation bar, making the backside of the navigation bar to show up. Figure 5-22 shows a sequence of images where a user changes into the immersive mode, moves around within the virtual scene, modifies the content with the manipulator tool, and then switch back to the AR mode.

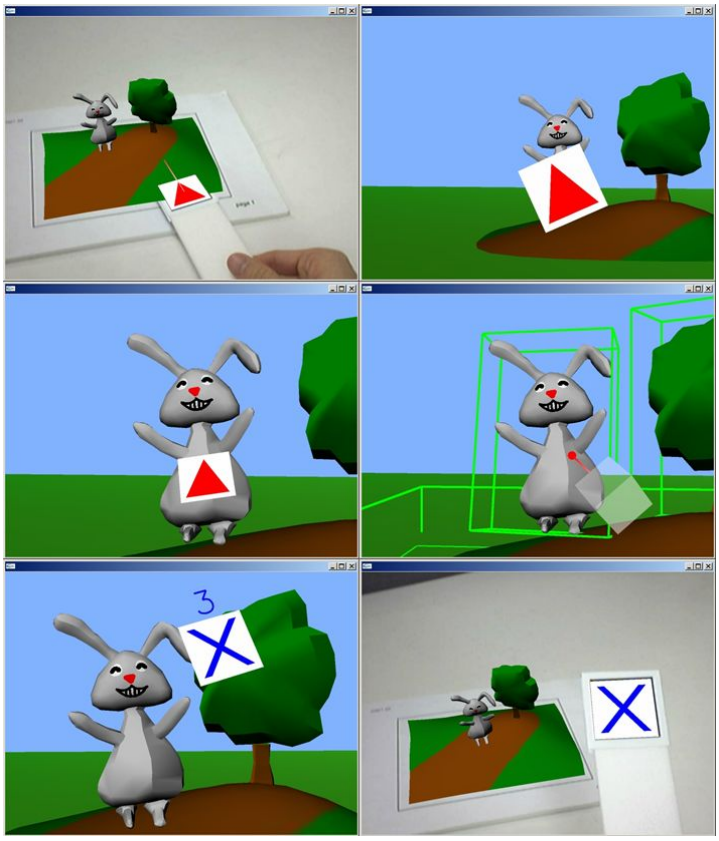

Figure 5-22 Switching between AR and immersive VR mode in iaTAR

The results of immersive authoring are needed to be saved into a file system. For this purpose, the authoring system provides the archive manager prop. The constructed virtual world can be saved into a XML styled script TARML (see Appendix A) for later

use. The scripts are represented as numbered archives in the archive manager, and the user can choose the archive number and save to or load from the selected archive (see Figure 5-23).

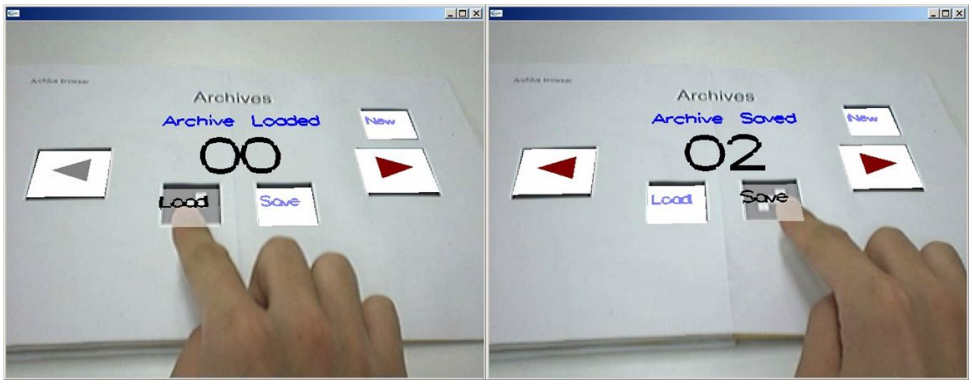

Figure 5-23 Loading and saving the constructed content in iaTAR

5.5.3. Implementation

The iaTAR system is implemented on a consumer level personal computer. The computer runs Microsoft Windows XP operating system on a Pentium 4 processor with 1GB main memory. A graphics card (nVIDIA GeForce4) with hardware 3D acceleration function is used for the OpenGL graphics processing.

The iaTAR system uses a custom 3D model loader and scene graph rendering engine, based on the OpenGL library, to visualize the 3D graphics contents and the virtual authoring tools. For tracking physical objects, we use a vision based tracking method. A plain USB web camera from Logitech acquires live video stream, and the ARToolKit [42] software library calculates the 3D position and orientation of the visual markers in the video image. The capturing resolution of the camera is set to 320x240 and the shutter speed is 30 frames per second. The camera is mounted on a head mounted display to provide a real world view to the user, forming a video see-through AR configuration.

For the button pressing interactions on a prop, the authoring tool uses a low cost AR interaction paradigm called the occlusion-based interaction, developed by the author, in which visual occlusion of physical markers (by user's fingers usually) are used to provide intuitive two or three dimensional interaction with tangibility. Figure 5-24 illustrates a simple application of occlusion-based interaction where a marker pattern occluded by the fingertip triggers a button-pressing event. The basic principle can be applied to design a variety of tangible interfaces such as menus, buttons, slider bars, keypad, and even 3D object manipulation. For a more detailed description of this technique, the readers are referred to [16].

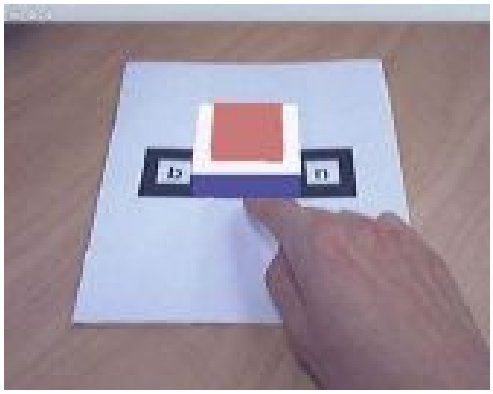

Figure 5-24 A Tangible AR button using the occlusion-based interaction method

To make the interaction easier for selecting components with the manipulator and inspector pads, bounding boxes are visualized around the component objects, and their colors are changed according to their status: green for normal, orange when pointed and red on selected. These bounding boxes and the logic box components are only shown when there are authoring props within the user's view.

Chapter 6. User Study

To validate the advantages of immersive authoring method, and to clarify the usability of implemented immersive authoring systems, user studies are unavoidable. Before we go into the details of the user study, we first indentify expected advantages of immersive authoring method. Expected advantages can be grouped into two: efficiency and effectiveness.

By efficiency, easiness in learning and using the method can be thought, as well as lesser authoring time. The feature of easy to learn and use the proposed method is especially important for the users with non-technical background. Since one of the main purposes of the immersive authoring is to offer the driving power in producing VR contents to artists and content producers, the proposed method must be easy enough to be learnt and used by those with non-technical background. Taking lesser authoring time is another representative measure of efficiency. Two main factors reduce authoring time in immersive authoring. One is by eliminating the time for switching between the authoring and testing environment, and the other is by taking advantages of direct manipulation in three-dimensional spatial tasks. The author performed a user experiment to validate these benefits and the results are described later in this chapter.

The quality of resulting virtual world contents produced by the authoring system is one of the representative indicators for its effectiveness (or productivity). One of the well-known quality measurements for virtual environments is ‘presence’. However, quality of a virtual world, including presence, is subjective measures and thus it is hard to provide objective measures. Moreover, while discussing on the effectiveness of an authoring tool, it is not enough to argue with a small number of cases and results from a short-term study. The productivity of an authoring tool might be recognized by the users

after having sufficient experiences using and working with the tool for creating numbers of practical contents, and this takes a long term of time. Unfortunately, there is no such comprehensive study available yet, however, in this thesis, the author presents some case studies, instead.

6.1. Authoring case study

Although it is hard to show numerous results from a lengthy study in this thesis, the author presents some cases of virtual world authoring experience that gives a glimpse on productivity and effectiveness of the proposed immersive authoring systems.

6.1.1. PiP

Two interactive virtual worlds have been built for testing and demonstrating the benefits of immersive authoring using PiP. One is the "Undersea World" in which various types of fishes swim and interact in different styles, and the other is an interactive VR story named "Romancing the Stone."

In the "Undersea World," the initial static scene was given by a quick editing of a PiP script, in which a number of non-moving fishes are scattered in the 3D space (this was done for convenience sake, even though pre-modeled objects can be created, destroyed and placed at different locations through 3D interaction as well). The two major objects in this virtual world were the fish and the food. There were four types of fishes, differing in their appearances with pre-modeled 3D geometric models (see Figure 5-7 to Figure 5-9). As also illustrated in the previous sections, the fishes were assigned with different behaviors through physical interaction, including the "eat-food-if-found," "move-forward-and-avoid-collision," "flee-upon-touch," and others. For instance, the "move-forward-and-avoid-collision" action was specified through setting a spatial context region in front of the fish, and making the fish turn away if another object was in that region. As aimed, all of the behaviors were modeled and tested iteratively within the

virtual environment without going back to use the usual 2D programming environment.

The second example illustrates how PiP can be effective in creating of interactive VR adventures. Starting with some basic pre-modeled objects and initial scenes, a first person viewpoint interactive adventure can be authored at a whim of the user very easily, as whatever pops into the director's mind. The virtual world named "Romancing the stone" is a simple interactive game, in which the player (the final user) is to overcome obstacles to acquire jewels, not get killed in the way, and reach the final destination (see Figure 6-1). Fifteen pre-made objects were used and four initial scenes were given as a start.

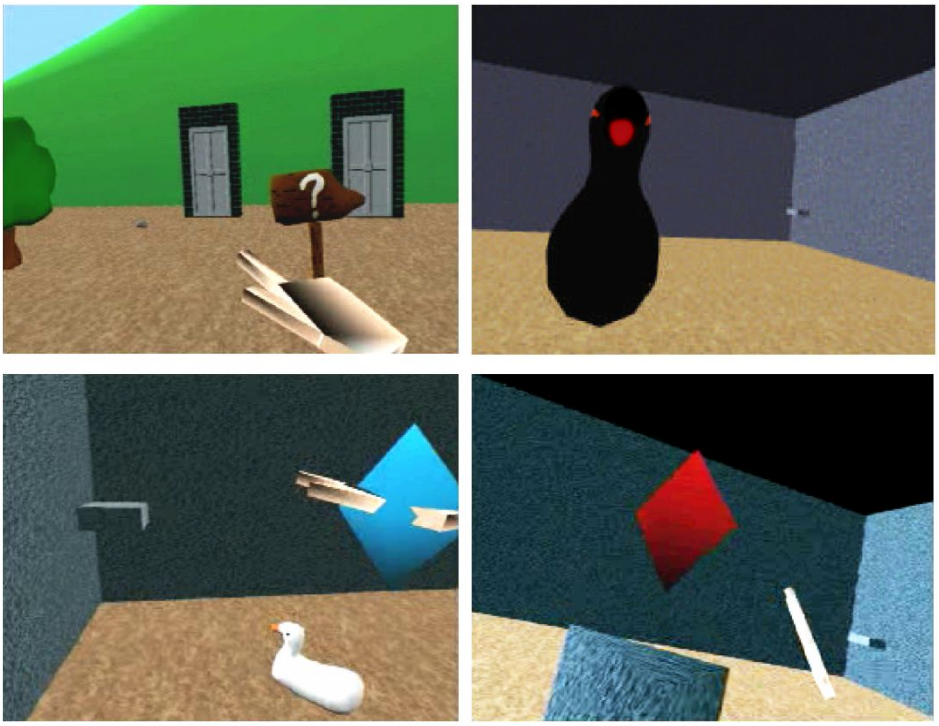

Figure 6-1 Scenes from the interactive VR adventure: "Romancing the stone"

In the first scene, the user faces two doors (upper left of Figure 6-1), one connected to a room with jewels and another to a pitfall with a monster in it, where the user is doomed to die. One of the typical behaviors in an interactive game is change of scenes upon user selection. The director can demonstrate this behavior by "touching" a door to specify the event that triggers an action to transport the user to the designated room. The transport action is given by using the menu items that allows the user to switch between different scenes.

When the player chooses the right door, the room with jewels appears (lower left of Figure 6-1). Inside the room, a number of jewels are scattered on the floor, and the player is to collect the jewels and place them at a specific place within the room in a given time limit. Ducks, wandering in the room, swallows the jewels, while the player can have the duck spit out the jewel by beating it with his (or her) hand. If the player accomplishes the given task in time, s/he finally enters the last scene where a big crown jewel is presented (lower right of Figure 6-1), otherwise, the player is descended into the pitfall and the game ends.

All object behaviors run concurrently in PiP, while timer events can be used to coordinate and synchronize different object behaviors. In this example, a timer event is set through the timer widget, available on the menu, so that upon this time event, an appropriate set of actions can be triggered to check if the user has accomplished the task and move on to the final presentation room.

It only took about an hour to author and validate the entire content, excluding the time for geometric modeling of the objects.

6.1.2. iaTAR

The first example content built with the iaTAR system is a simple scene with a windmill (see Figure 5-19). The scene consists of three virtual objects: the ground, a tower and a vane. It took about a minute to place the virtual objects and check that every thing was placed in the right place. A logic box representing a rotation behavior was used for specifying the vane to spin around. It took less than 3 minutes to construct the whole scene, connect the properties to define the behavior, and to validate the final product.

In the second example, interactive features were added to the content. Figure 6-2 shows a sequence of images, constructing an interactive Tangible AR content similar to the Shared Space application [3]. This example content shows two tiles with a virtual object on each, a hare and a tortoise. The user can examine the virtual objects by manipulating the tiles on which they are anchored. When two tiles are brought close together, the scene is changed and the hare and the tortoise greet each other (see the last row of Figure 6-2).

To build this content, four virtual objects were needed: the normal and greeting posed models for the hare and tortoise. First, the virtual objects were placed on two physical tiles, one for the hare and another for the tortoise. The visibilities of the virtual objects were controlled by the proximity value of the physical tiles. In order to check the distance and to control the visibilities, we used a logic box with a function calculating proximity. The logic box had two input properties of position, and Boolean valued output properties, named ‘near’ and ‘far’, representing whether the two input positions were close enough or not. By connecting the position properties of the two tiles to the logic box input, and the ‘near’ and ‘far’ properties of the logic box to four virtual objects’ visibility, properly, the authoring was completed. Only about 5 minutes were needed for building and testing the whole content.

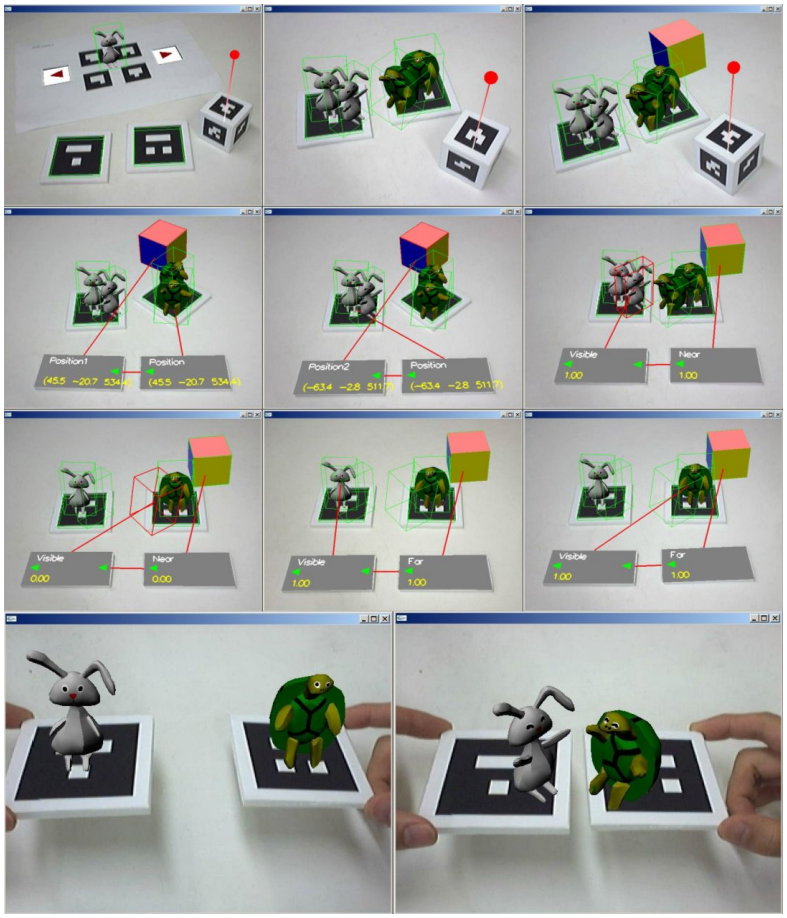

Figure 6-2 Authoring an interactive card matching content in iaTAR

The last example content authored with the iaTAR system is an AR storybook application, similar to the MagicBook [4] (see Figure 6-3). The story for this content was chosen from one of the popular stories of Aesop, 'The race between a hare and a tortoise.' The story consists of three main scenes: starting the race, the hare taking a nap and the tortoise winning. To add interactivity to the story line, we made a decision point on the second scene to let the users choose whether the hare should sleep or not. According to the user's decision, the winner on the last scene would be determined differently.

Thirteen pre-modeled 3D objects were brought in and three sheets of paper with printed markers were used as the book pages. To implement the interactive feature, the director used occlusion-based interaction properties of physical object components: a set of Boolean valued properties indicating which button (i.e., marker) was pressed (for the last). These properties were connected to the visibility of the characters placed on the final scene, selecting different endings of the story according to the user's decision. It took about 15 minutes to construct the scenes and to connect object properties for implementing the interaction.

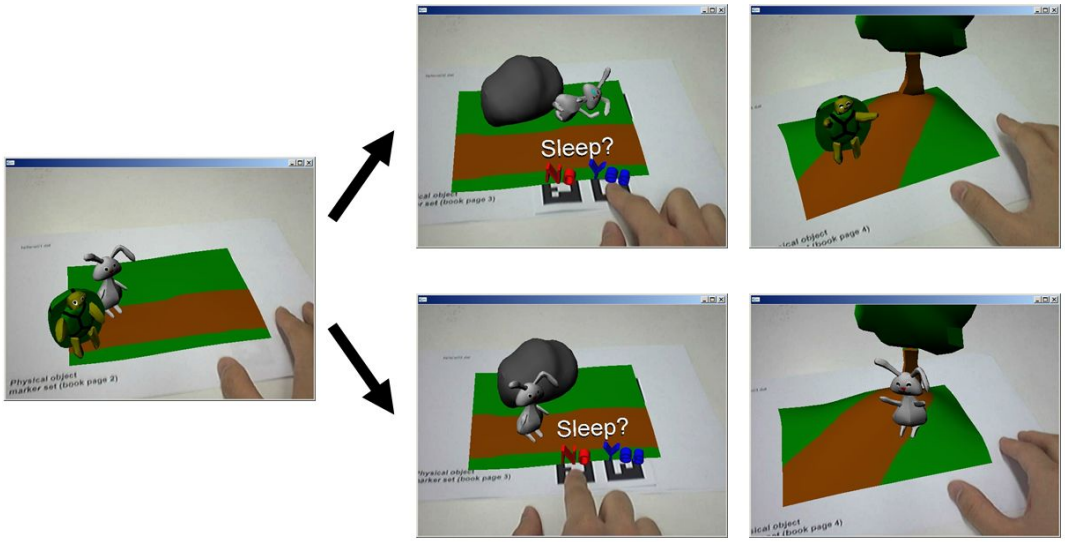

Figure 6-3 Tangible AR interactive storybook

6.2. User experiment

To validate the postulated advantages of immersive authoring, the author performed a formal usability experiment. The main purpose of the experiment was to show efficiency of the immersive authoring method in comparison to conventional development methods. Under this purpose, the experiment was designed to collect performance data while subjects author immersive content according to the given specifications. The immersive authoring tool used for the experiment was the iaTAR system, and for the counterpart conventional development method, the author chose scripting on a desktop computing environment. TARML, an XML based script language, was used for representing Tangible AR content with scripts. The full specification of TARML is available in Appendix A.

While 2D GUI tools were also considered as a counterpart, there were no such tool available, yet, which uses the same virtual world model with iaTAR; the author concluded that the authoring tools under comparison might, at least, have the same content model, so that the experiment could be balanced. Although 2D GUI tools are not directly compared to iaTAR in this thesis, the author notes that a previous study [17] with informal direct comparisons has partially demonstrated similar results with this experiment.

In this experiment, our main hypothesis is that the immersive authoring would be significantly more efficient, with less overall authoring time (particularly for tasks with spatial impact), and be more natural and easy to use compared to the desktop counterpart. To put it more precisely, the followings are the three hypotheses we are investigating in this experiment:

Hypothesis I. Immersive authoring method is significantly efficient in authoring immersive contents in comparison to conventional development method.

Hypothesis II. Immersive authoring method is significantly more efficient on spatial authoring tasks than on non-spatial tasks.

Hypothesis III. Immersive authoring method is significantly easier to use and learn in comparison to conventional development method.

6.2.1. Experimental design

The main structure of the experiment was designed as a two-factor within-subject experiment. The independent variables were the type of the authoring system used (iaTAR or TARML) and the type of authoring tasks involved (spatial and non-spatial). Authoring with the iaTAR system represented the use of immersive authoring method, and scripting with TARML represented the use of a conventional development method. The major dependent variables were the task completion time and the usability survey answers. The task assigned to the subject was to construct content that satisfied a given set of requirements, using the particular authoring system. The subjects were given as much time to finish the task as needed, but were instructed to complete the task as quickly and as accurately as possible.

The experimental procedure for a participant consisted of two sessions: the training session and the experimental session. During the training session, which took about an hour, participants learned the basic concepts of Tangible AR content and the respective authoring methods using iaTAR and TARML. Note that the syntax for the TARML scripts was sufficiently simple and easy to learn and use, especially for our subject group who had engineering backgrounds (see section 6.2.2 for detail information on the subject group). In addition to the detailed briefing, the subjects practiced with each authoring tool by constructing sample Tangible AR content.

In the experimental session (which followed the training session after a short break), the

subjects had another practice trial of authoring Tangible AR content, in order to help them recall the authoring methods after the break. After the practice trial, the subjects were asked to author six different types of Tangible AR content according to a given specification using both authoring methods. The order of the authoring methods used and the order of the six types of contents for each participant were counter-balanced. The experimental session took approximately one and a half hours. The overall experimental process is shown in Table 6-1.

Table 6-1 Experimental procedure

Sessions	Detail tasks	Time
Training (1 hour)	Subject feature survey questionnaire	5 min.
	Instruction: Overview of Tangible AR content	10 min.
	Instruction: Tutorial on TARML	5 min.
	Practice: TARML	15 min.
	Instruction: Tutorial on iaTAR	5 min.
	Practice: iaTAR	15 min.
	Post-training survey questionnaire	5 min.
break	Take a rest	5 min.
Experiment (1.5 hour)	A practice trial for both iaTAR and TARML with a particular content specification	Approx. 90 min.
	6 trials for both iaTAR and TARML with particular content specifications	
	Post-experiment survey questionnaire	

During the experimental session, to prevent any possibility of the subjects being unduly guided by the experimenters in any way, the subjects were not allowed to ask questions about how to use the authoring interfaces. Instead, the participants were allowed to use the user guide document freely for reference, which was provided at the training session. The experimenter was allowed to help the subject only when the subject got "lost" (e.g., not knowing what to do even after looking up the user guide) and it was determined that they had spent too much time on a particular subtask, i.e., more than one minute. In most cases, users knew exactly what to do, but had forgotten the exact syntax (for TARML editing), or how to perform certain operations (for immersive authoring). One minute was deemed approximately the right amount of time, based on our prior observation, to resolve such a problem in authoring. Any extra time spent due to help was subtracted from the task completion time. In this way, a provision was made so that the overall performance data was not severely biased by a few such outliers.

The requirement specification sheet was provided and explained by the experimenter at the beginning of each trial. The subject was asked to build the content described in the requirement specification sheet as quickly and accurately as possible. The subjects were allowed to refer to the requirement specification sheet and to ask questions to the experimenter about it whenever needed. In addition, the experimenter also periodically read out the requirements for the participant to remind the subject what to do next. Such a procedure was needed because when using the immersive authoring method, the subjects, wearing the head mounted display (HMD), sometimes could not read the specification document, due to the narrow field of view and low resolution of the display used.

There were six experimental trials, each with a different requirement specification sheet. Table 6-2 shows two example requirement specifications, one spatial and the other non-spatial, used in the experiment. In each trial, participants used both immersive and script editing authoring methods, in order to build the same content twice. Three trials

included only spatial tasks, and the other three trials only non-spatial tasks.

Table 6-2 Sample requirement specifications used in the user experiment

Spatial tasks	Non-spatial tasks
0. Use page1 to construct a scene on. 1. Place the terrain (Models/underhill.obj) on page1. - The terrain must be parallel to page1 - The hilltop must be on the right side. - Scale it up to fit the size of page1. 2. Place the tree (Models/tree.obj) - Place it on the hilltop. - Be sure the tree is neither floating over nor buried under the ground. - The flat side of the tree must face to the road on the hill. 3. Place the rabbit (Models/ rabbit.obj) and the turtle (Models/ turtle.obj) - Both of them are at the left front end of the road, under the hill - The rabbit is on the left to the turtle - Both of them are facing front while slightly turned to face each other.	0. A scene will be provided including the following objects. - Two physical cards (Markers/card1.dat and Marker/card2.dat) are used. - Two models of rabbit in different poses are placed on the first card and two models of turtle in different poses on the other card. - A logic object with a function of checking distance is provided. 1. Connect the 'position' properties of both cards to 'input position 1' and 'input position 2' properties of the logic object. 2. Connect the 'far' property of the logic object to the 'visible' property of the rabbit standing upright. - Do the same to the turtle standing upright. 3. Connect the 'near' property of the logic object to the 'visible' property of the rabbit with its hands up. - Do the same to the turtle with its hands up. 4. Change the value of the 'threshold' property of the logic object into 100.

In the trials with spatial tasks, the participants were asked to build an AR scene, such as in Figure 6-3, positioning and orienting four virtual objects, and scaling one of them. To help the subjects make use of their spatial perception, spatial properties (i.e., positions, orientations and scale factors) were described in a relative manner, and not by specific

values or numbers. In addition, a sample picture of the final scene was shown to help the subject understand and remember the scene to be constructed. In the trials with non-spatial tasks, the subject began with a particular pre-modeled virtual scene. Using the virtual objects in the particular virtual scene, the subjects were asked to make dataflow links and to change object properties to a specific value. As it was with the spatial task specifications, the task amounts were also balanced between the non-spatial task specifications; each specification included six dataflow links and one property value to be set.

To measure the user performance, task completion time was recorded for each trial, for each authoring method. The number of times the participant referred to the user guide, and the number of times the participant got lost with the interface were also counted. The period of time the subject got lost (looking at the user guide for more than a minute and/or not knowing what to do) was subtracted from the task completion time, as already mentioned previously.

Subjective measurements were also collected with questionnaires at the end of the training and experimental session. At the end of the training session, the participants were asked to rate how easy the given authoring method was to learn. At the end of the experimental session, they were asked to rate how easy the given authoring method was to use, and how confident they were with the content they built using each authoring method. Ratings were given on a Likert 7-point scale (0: very difficult/unconfident, 3: neutral, 6: very easy/confident). Other subjective opinions, such as user preference and the strengths and weaknesses of the authoring methods were also collected.

6.2.2. Experimental setup

The experimental environments for script editing and immersive authoring are shown in Figure 6-4. A desktop computing environment (using a 2D display, keyboard and mouse)

was provided for the script editing method. Because the target content to be built was an AR based content, a camera was set up on a fixture stand for testing the content script. The user was allowed to change the camera location freely if needed. For the immersive authoring configuration, we used the iaTAR system. The interface of the iaTAR system consisted of an HMD with a camera attached in front, in order to provide a video see-through functionality, and tangible props that were used as authoring tools.

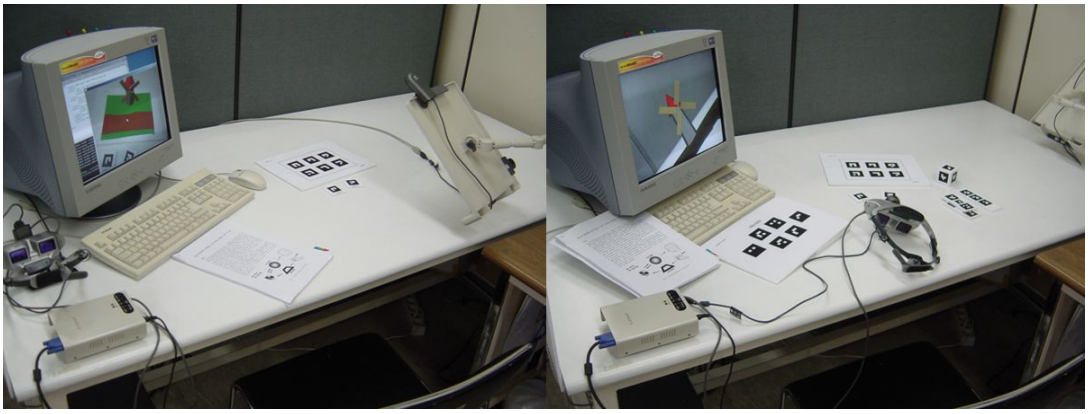

Figure 6-4 Experimental environments: script editing (left), immersive authoring (right)

While one can also edit a script within an immersive environment, we used a conventional desktop computing configuration to avoid usability problems that might occur when using a HMD interface, which usually has low video resolution and narrow field of view for displaying text information. Until now, conventional desktop computing environment is known to be the best configuration for manipulating text information, and therefore we chose it instead of other configurations that are not verified and not common.

Twelve graduate and undergraduate students with engineering backgrounds participated in the experiment. The age of the participants ranged from 19 to 26 and they all had sufficient typing skills for script editing (typing an average of 242 characters per minute). Half of the participants were taking the VR class at the computer science department.

Thus, they had brief experiences (approximately 3 months) in VR system development. The other half did not have any experiences on virtual reality system development nor 3D graphics programming. The two groups showed no statistically significant difference in age and typing skill with ANOVA test ($\alpha = 0.05$), although, the group taking the virtual reality class turned out to have more experience in using programming languages (C/C++ or JAVA). Table 6-3 summarizes the features of the participants.

Table 6-3 Features of subjects in the user experiment

Features	All subjects	VR class subject group	non-VR class subject group	ANOVA between groups
Number of subjects	12	6	6	-
Gender	All male	All male	All male	-
Age (years on average)	22.2 (SD=2.04)	22.5 (SD=1.76)	21.83 (SD=2.40)	F(1,10)=0.301 p = 0.5954
Experience with C/C++ or JAVA (years on average)	2.8 (SD=1.80)	4.3 (SD=1.03)	1.25 (SD=0.61)	F(1,10)=39.566 p < 0.0001
Experience with HTML (years on average)	2.2 (SD=2.04)	3 (SD=2.45)	1.3 (SD=1.21)	F(1,10)=2.232 p = 0.1660
VR (or 3D graphics) system development experience	-	All about 3 months	All not at all	-
VR system (or 3D game) experience	-	All more than 5 times	Varies from not at all to more than 5 times	-
Typing skill (characters per minute on average)	242.4 (SD =78.89)	268.7 (SD=84.19)	216.2 (SD=70.35)	F(1,10)=1.374 p = 0.2683

* SD: standard deviation

6.2.3. Experimental results

In order to investigate the Hypotheses I and II, we have conducted a series of one-way within subjects ANOVA for comparing users' task performances between iaTAR and TARML. All of the tests were held with an alpha level of 0.05. While the two hypotheses could be investigated with a single two-way ANOVA test on task completion time over two independent variables (i.e., the authoring tools and task types), this was avoided since it would introduce biased results due to the differences in the amounts of task carried out. That is, the task amounts were only balanced between the same task types, and thus it would be unfair to compare the task completion time, directly, between spatial tasks and non-spatial tasks. As an alternative, we introduce an efficiency factor for the immersive authoring method, and compare this over different task types with a separate one-way ANOVA test.

First, to validate the overall efficiency of the immersive authoring (Hypothesis I), the total task completion time for authoring all six types of contents using each method was compared to each other. The average total authoring time spent using iaTAR was 27 minutes 43 seconds, and for TARML, 38 minutes and 37 seconds. The ANOVA revealed a statistically significant difference with $F(1,11) = 37.20$ and $p < 0.0001$. That represents about 28% of time saving when iaTAR is used in comparison to TARML. According to this result, we can conclude that the Hypothesis I is valid. See appendix B.1 for detail results of the ANOVA test.

As described earlier, in order to assess if iaTAR is indeed more efficient for spatial tasks (Hypothesis II), we compared the efficiency factor of the immersive authoring method according to the task group: spatial and non-spatial. The efficiency factor ***E*** of iaTAR (over TARML) for a given task was defined in the following way:

$$E(task) = T(iaTAR, task) / T(TARML, task)$$

where ***T(x, y)*** is the time spent for completing task ***y*** with the authoring tool ***x***. We applied ANOVA with the efficiency factor for each task group as the dependent variable. As a result, the efficiency factor of iaTAR over TARML turned out to be significantly greater ($F(1, 11) = 115.2647$; $p < 0.0001$) for the spatial task (mean = 0.5060; standard deviation, SD = 0.1130) than for the non-spatial (mean = 1.1764; SD = 0.2461). Hence, we can conclude that the Hypothesis II is valid as well.

Our results so far appear to indicate that iaTAR is more efficient for spatial tasks than for non-spatial tasks. To confirm this further, we investigated if the actual amount of time that iaTAR spent on non-spatial tasks was significantly more than that for spatial tasks: we directly compared the task completion time of iaTAR over TARML for each task group.

First, we compared the task completion time between iaTAR and TARML, only with the spatial tasks. As expected, the iaTAR clearly took shorter amount of time for the spatial authoring tasks (see Table 6-4). The total time spent for completing the spatial authoring tasks using TARML (25 minutes 12 seconds on average) was approximately twice as much as that of when using iaTAR (12 minutes 13 seconds on average), which was a significant difference under the ANOVA test ($F(1,11) = 99.94$; $p < 0.0001$).

However, for non-spatial authoring tasks, iaTAR turned out to be inefficient compared to TARML (see Table 6-4). With iaTAR, it took about 13% longer for subjects (15 minutes 12 seconds on average) to complete the non-spatial tasks, compared to TARML (13 minutes 24 seconds on average), and the difference was statistically significant ($F(1,11) = 6.20$; $p = 0.0301$). While iaTAR took longer with the non-spatial authoring tasks, the difference with TARML was not as large as that of the case with the spatial authoring task where iaTAR outperformed twice over TARML as mentioned earlier. We discuss this result further in section 6.2.5. Figure 6-5 shows the total tasks completion time according to the task groups in a graph. See appendix B.2 for detail results of the ANOVA test.

Table 6-4 Experimental result: Total task completion time

	iaTAR	TARML	ANOVA
Total task completion time	27:43 (SD = 5:10)	38:37 (SD = 7:44)	$F(1,11) = 37.20$ $p < 0.0001$
Completion time for spatial tasks	12:31 (SD = 2:50)	25:12 (SD = 4:47)	$F(1,11) = 99.94$ $p < 0.0001$
Completion time for non-spatial tasks	15:12 (SD = 2:37)	13:24 (SD = 3:28)	$F(1,11) = 6.20$ $p = 0.0301$

* Time in 'minute:second' format. SD: standard deviation

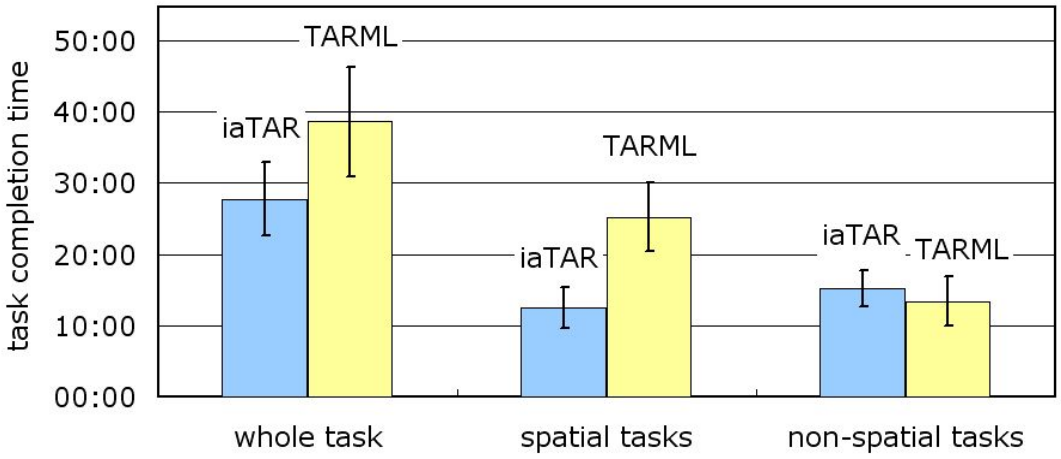

Figure 6-5 Experimental result: Total task completion time

While we only described the comparison of total task completion time for assessing Hypothesis I and II, so far, the comparison of the average task completion time of all six trials showed similar results, as well. The iaTAR showed better user performance by saving 28% of average task completion time; the average task completion time for TARML was 6 minutes 26 seconds (SD = 2 minutes 40 seconds), while for iaTAR was 4 minutes 37 seconds (SD = 1 minute 22 seconds). The results on the difference between

two authoring methods for each task group were similar to the results with total task completion time, as well. With spatial tasks, iaTAR out performed about twice in comparison with TARML, and for non-spatial tasks iaTAR took slightly more time. The results are summarized in Table 6-5. See appendix B.3 for detail results of the ANOVA test.

Table 6-5 Experimental result: Average task completion time per trial

Task group	iaTAR	TARML	ANOVA
Spatial tasks	4:10 (SD = 1:14)	8:24 (SD = 2:00)	$F(1,10) = 132.28$ $p < 0.0001$
Non-spatial tasks	5:04 (SD = 1:12)	4:28 (SD = 1:32)	$F(1,10) = 14.86$ $p = 0.0032$

* Time in 'minute:second' format. SD: standard deviation

To assess the usability (i.e., ease of use and learning, namely the Hypothesis III), we compared the subjective ratings collected with the questionnaire asking how easy iaTAR or TARML was to learn and use. Based on the results of subjective ratings, no statistically significant differences were found between iaTAR and TARML ($\alpha = 0.05$; see Table 6-6). For instance, iaTAR was only marginally easier, strictly speaking, to learn than TARML, with the p-value = 0.067. However, according to the results of debriefing, this appears to be mainly because of the relatively low quality of the device and sensing accuracy rather than from the method itself (see section 6.2.4). On the other hand, while the syntax of the language might have taken the users some time to learn, the interface for script editing (i.e., desktop interface) was already quite familiar to most users. See appendix B.4 for details of the ANOVA test result.

Table 6-6 Experimental result: Subjective ratings

Questions	iaTAR	TARML	ANOVA
Easy to learn	4.50 (SD = 1.168)	3.92 (SD = 1.240)	F(1,11) = 4.11 p = 0.0674
Easy to use	4.08 (SD = 0.996)	4.25 (SD = 1.138)	F(1,11) = 0.17 p = 0.6887
Confident with authoring results	3.92 (SD = 1.165)	4.50 (SD = 0.905)	F(1,11) = 3.01 p = 0.1106

* Ratings in Likert 7-point scales (0: very difficult/unconfident, 3: neutral, 6: very easy/confident)
* SD : standard deviation

In order to assess the Hypothesis III further, we compared the number of times subjects referred to the user guide document. Subjects referred to the user guide document 3.9 times on average (SD = 3.55) when using TARML, whereas they never needed to do so for iaTAR. Subjects with VR backgrounds only used the guide 1.8 times (SD = 1.33) on average, whereas those with no VR background used it 6 times (SD = 3.95). This demonstrates the obvious fact that there is a mental skill factor involved with using the conventional programming approach. It is desirable to eliminate such an obstacle as much as possible for non-technical content developers such as artists.

Another indication of the authoring efficiency, or ease of use and learn, can be observed from the number of times the subjects got lost when using the respective authoring methods. Two users (both not from the VR class) got lost two times each when using TARML. They reported that they had gotten lost due to the confusion with the TARML grammar and problems with perceiving rotations in the three-dimensional space.

In order to investigate another aspect of usability, the author compared the performance according to the participants' background knowledge (i.e., the virtual reality development

experience). The average of task completion time for each trial was used as a dependent variable, and for the independent variables the author used the authoring tool and the group of participants (one group those who took the virtual reality class and the other who did not), both in nominal scale (category variable) with two levels. Since the trials with spatial tasks and non-spatial tasks were not balanced in their task amount, the results of the analysis are collected separately between two task types. Figure 6-6 shows the average task completion time per trial according to the authoring methods for each subject group. No statistically significant difference ($\alpha = 0.05$) was found between the subject groups with regard to the task performance for both spatial and non-spatial authoring tasks. This result can be thought as another indication that iaTAR is easy enough to learn and use for those without background knowledge on VR technologies and 3D authoring tasks.

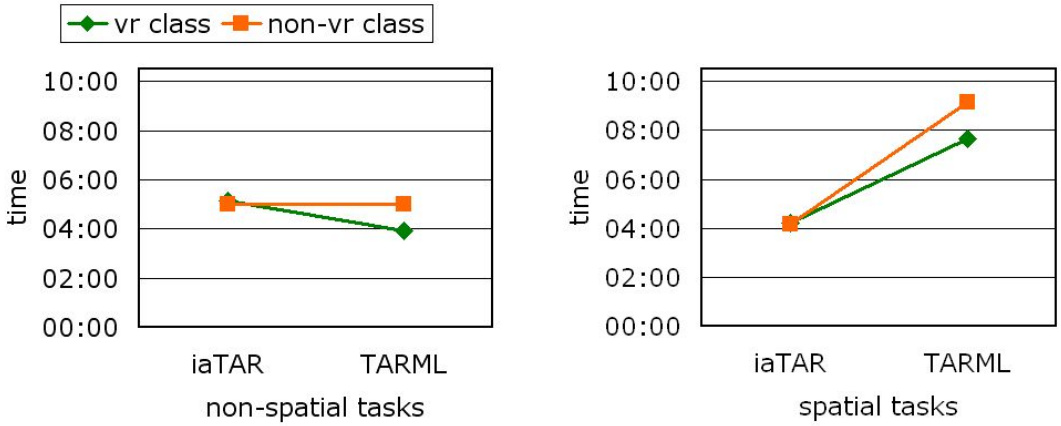

Figure 6-6 Comparison of average task completion time between participant groups

Besides, with non-spatial tasks, a statistically significant interaction between the subject group and authoring methods was found ($F(1, 10) = 16.37$; $p = 0.0023$). From this, we can posit that the background knowledge on VR development gave an advantage when using the conventional authoring method (i.e., TARML). We believe that this resulted from their prior expertise and experience in content development, especially dealing with

3D transformations and describing them with a computer language (such as scripts). And although it did not appear statistically significant ($F(1,10) = 4.45$; $p = 0.0611$), a similar trend was found with the spatial tasks as well. See appendix B.3 for detail results of the ANOVA test.

6.2.4. Debriefing

At the end of the questionnaire, participants were also asked to write down their opinions on strengths and weaknesses of each method, freely. The reported opinions are summarized in Table 6-7 with a number of participants who mentioned it. Although it is hard to say their statistical significance (since they were collected in informal descriptions), the results briefly suggests some recognizable points that should be considered in immersive authoring systems.

According to the reports, the most prominent strength of iaTAR turned out to be its easiness in placing and manipulating virtual objects in three dimensional space, while most (actually all, except the top most mentioned one) of the reported weakness of TARML related to its tiresome trial-and-errors during placing virtual objects. On the other hand, the top two of the reported strengths of TARML were related to its prominence in assigning exact numerical values to object properties, while the unavailability of fine positioning (implying assigning exact values to the properties related to their pose) were mentioned most as a weakness of iaTAR. Other minor opinions also suggested some valuable points for improving the iaTAR system, such as introducing undo and copy-paste functions.

On the question asking their inconvenience while using iaTAR , most of the participants answered the narrow field of view of the video see-through head mounted display, suggesting that there are needs for improvements in hardware devices. Inconvenience with the repeated trial-and-errors was mentioned most in using TARML, which is hard to

make improvements due to its nature. However, the situation might be improved a bit if the users could edit scripts on a head mounted display (leaving out its usability problems) and check the results instantly, while this is also considered as one of the approaches in immersive authoring, the virtual desktop interface approach. Table 6-8 summarizes the answers from participants.

Table 6-7 Debriefing results: Strengths and weaknesses of each method

Questions	Answers	Number of participants
Strengths of iaTAR	Easy when placing virtual objects	7 out of 12
	Easy to learn	5 out of 12
	Can check the results instantly (concurrently)	3 out of 12
Weaknesses of iaTAR	Coarse object positioning (fine tuning not available)	7 out of 12
	Tracking lost	3 out of 12
	Narrow field of view	2 out of 12
	Inconvenience in changing property values	2 out of 12
	Undo function unavailable	1 out of 12
Strengths of TARML	Easy to assign exact values to the property	6 out of 12
	Fast when given exact specification values	4 out of 12
	Copy and paste function available	1 out of 12
	Easy syntax	1 out of 12
Weaknesses of TARML	Need to learn grammars (relatively not easy to learn)	6 out of 12
	Repeated trial-and-errors when placing objects	5 out of 12
	Need to reload the script after editing	2 out of 12
	Hard to perceive rotations in three-dimensional space	1 out of 12

Table 6-8 Debriefing results: Inconvenience with each method

Questions	Answers	Number of participants
Inconvenience with iaTAR	Narrow field of view	10 out of 12
	Unclear focus with HMD	4 out of 12
	Depth perception problem	1 out of 12
Inconvenience with TARML	Repeated trial-and-errors	6 out of 12
	Three-dimensional space perception	1 out of 12

Finally, the participants were asked with their preference on the authoring methods. None of them chose to use only one method, but they chose to use them together. They mentioned that direct manipulation with iaTAR would be useful with spatial tasks, such as constituting the overall scene, while editing TARML script would help with fine tuning the property values and other logical (or non-spatial) tasks such as linking objects.

6.2.5. Discussion

According to the results of the user experiment, we can conclude that there are significant evidences for the Hypotheses I and II. That is, using the immersive authoring tool was efficient in comparison to using conventional development method, with approximately 30% reduction in time to complete the given authoring task. This was because immersive authoring yielded twice the performance in comparison to the conventional authoring method for spatial tasks, while exhibiting comparable performance for non-spatial tasks.

Also note that, in an actual use of the conventional desktop authoring method, the user would have to change to an "execution" mode and "wear" the system (e.g., wear the HMD and the associated sensors) to test the resulting content, which is a time consuming and bothersome process. Such a factor was not even included in assessing the non-immersive

authoring time. Thus, it can be predicted that the overall authoring and testing time will be much shorter with immersive authoring, even for the non-spatial tasks.

Finally, iaTAR demonstrated no marked difference in ease of learning and use in comparison with TARML editing, which employs one of the most familiar interfaces, 2D editing (a similar argument can be extended for 2D GUI also). Although the results of subjective ratings demonstrated no statistical difference between the two, the number of times participants referred to the user guide document and the number of times participants got lost during the experiment provide indications of higher usability for iaTAR even for non-logical tasks. Noting that the most inconvenience in using the iaTAR was caused by imperfect devices (e.g., wired sensors and low resolution HMD), we anticipate that higher quality equipments will significantly improve the usability of iaTAR.

Subject debriefing also reflected the preceding analysis results. Although these informal results do not have any statistical significance, they still raise enough important issues, in the context of the quantitative results, to be considered when designing and evaluating immersive authoring systems. Most subjects reported that the ease of configuring objects in 3D space was the foremost merit of iaTAR and the ability to specify exact values was the foremost merit of TARML. The users of iaTAR complained about the standard AR related usability problems, such as the narrow field of view, degraded depth perception, the difficulty with continuous tracking and the inability to specify exact values. Consequently, the subjects expressed their preference in a specific manner. They preferred iaTAR for spatial authoring tasks, and TARML for the non-spatial and fine-tuning tasks.

While authoring of most VR-based content will involve both spatial and non-spatial (i.e., logical) tasks, their relative proportion will be different every time, depending on the particular target content. One can envision the authoring tool incorporating both

immersive and desktop authoring features, which can be used in a flexible manner depending on the nature of the target content. That is, for authoring spatial behaviors, the system provides immersive authoring interfaces, and for specifying non-spatial features, conventional desktop interfaces are available, as well. An alternative is initially performing design and implementations of the content, with the spatial aspects only vaguely specified using text input and 2D GUI, and then completing the final design and validation in immersive mode. It is also possible to convert logical tasks into spatial tasks, as suggested with the metaphorical object approach and the programming by demonstration approach. Moreover, as VR interfaces advance in the future, it would become also possible to manipulate virtual desktop interfaces efficiently within an immersive environment, just as we use 2D GUI fluently, today.

Chapter 7. Conclusion

Starting from introducing the concept of 'immersive authoring' of virtual worlds, in this thesis, the author suggested three approaches to achieving the proposed purpose: using virtual desktop interface, metaphorical virtual objects and programming by demonstration technique. The author analyzed the requirements of immersive authoring arranging them into design principles and the authoring task model. Appropriate interactions and interfaces were chosen and designed accordingly, and were implemented into immersive authoring systems. The content authoring experience with the implemented prototype systems showed how immersive authoring systems could be practically used in VR content development. Moreover, a formal user experiment provided solid evidences on the benefits and efficiency of immersive authoring method. Immersive authoring hinges on the advantages of merging the execution and the authoring platforms and the nature of 3D multimodal interfaces and immersion for producing higher quality VR-based contents in an efficient way.

7.1. Contributions

The most prominent advantage of using immersive authoring method, in comparison to conventional development methods, is merging of development and testing environment. Spatial and mental gaps, that content developer had to suffer with switching between two separate environments, are certainly reduced by removing the distinction between the authoring and the execution environment. Thus, a large amount of time and efforts spent for switching during development can be saved and invested in actual 'authoring' tasks.

Merging the two environments leads to another distinctive benefit of providing instant (or even concurrent) evaluation, which the newly proposed term 'WYXIWYG' represents. Fast evaluation of the resulting product has significant profits with virtual world

development, where various combinations of different content elements are actually explored during the development. While trying out a number of different configurations, fast evaluation of resulting virtual world is necessary to save time and efforts. Moreover, since VR contents are usually evaluated with a comprehensive subjective measure (i.e., presence), directors need to actually experience the resulting virtual world for every combinations, and immersive authoring certainly supports this.

Finally, by introducing direct 3D manipulation as a main interaction policy, immersive authoring becomes easy enough to learn and use for those with no background knowledge on computer programming and other technical stuffs. This offers the main seat of VR content production to the artists, special domain experts, and even to the end-users, who really are the natives of content production.

Gathering all of these contributions together, immersive authoring comprehensively contributes toward spreading VR-based contents to the market, and moreover, providing foundations for the success of VR technology as a next generation, future main stream of media industry.

7.2. Future research direction

Although the author described instances of virtual world model for practical use in this thesis, more researches are required to complete a comprehensive model for describing generic structures and behaviors of virtual worlds and virtual objects. Moreover, to make an immersive authoring system immune to the changes in virtual world models, component oriented programming techniques [35], such as CORBA, COM/DCOM, and JavaBeans, can be applied. With virtual world components implemented in a component software form, those modules can be dynamically binded on runtime, making possible to update the source code of each module, recompile them and reload without halting the main virtual reality (authoring) system running. In brief, advanced researches on virtual world models

would help developing more comprehensive immersive authoring tools, so that they could be used in practice.

Introducing collaboration (with other users) into immersive authoring system is also one of the interesting subjects for future research. When developing a software system, separating it into several software modules, programming them separately, and integrating them together into a complete system is a classical method when developing large-scale software. In advance to this offline process, 'extreme (or pair) programming' [2][44][37], one of the latest software engineering methods, lets a pair of programmers work on a single software code together. A group of programmers view and modify the same source code together at a same time, discussing and collaborating throughout the development process, and it is known that this improves not only the enjoyment of the problem solving process, but also the performance [22]. This flow of collaborative software development methods encourages the idea of collaborative immersive authoring of virtual environments. VR content directors need to work with other experts and staff members such as 3D graphics designers, application field experts and usability engineers, in addition to the system engineers, and for this purpose, advanced interaction models and associated system functionalities will be required to support such a collaborated effort of content creation.

Although there still exists some problems to be solved, according to the results shown in this thesis, the author is convinced of 'immersive authoring' as a next generation authoring method for VR environments. Being improved by further researches, the immersive authoring method will undoubtedly provide solid assistance to the VR technology (including AR), helping it to grow as a main stream of future media and human computer interface.

Appendix A. TARML specification

TARML is a markup language for describing Tangible Augmented Reality contents. The language is in XML style.

tarml

Header and tail of tarml file. All other tags are describe between these tags

<tarml>
    object-and-link-list
</tarml>

object-and-link-list a list of object and/or link tags

property

For describing the value of a property of an object. This tag is only used within object tags: pobject, vobject, sobject and lobject.

<property name="*property-name*" value="*value-string*"/>

property-name name of the property in string
value-string value of the property to assign. (string formatted according to its data type as below)

Data type	Format
Scalar	A decimal numeric value (ex) 10 123.45
Boolean	A decimal numeric value. 0 for FALSE 1 for TRUE. (ex) 0 1
Vector	A triplet of decimal numeric values delimited by blanks (ex) 100 123.4 45
Matrix	A Twelve-let of decimal numeric values in a row major order, delimited by blanks. (ex) 0.0 0.1 0.2 0.3 1.0 1.1 1.2 1.3 2.0 2.1 2.2 2.3

	represents the matrix: 0.0 0.1 0.2 0.3 1.0 1.1 1.2 1.3 2.0 2.1 2.2 2.3
Object name	A string (ex) marker01 car

pobject

For declaring a physical object.

<pobject name="*object-name*" marker="*marker-pattern-file-name*">
property-list
</pobject>

object-name name of the object in string
marker-pattern-file-name file name of the marker pattern or multi-pattern data file
property-list a list of property tags (usually no use since the whole properties are read only)

(ex)

```
<pobject name="page01" marker="Markers/page01.dat">
</pobject>

<pobject name="card02" marker="Markers/card02-40.patt">
</pobject>
```

Properties of pobject

Name	Data type	Read/Write	Description
ID	Scalar	Read	ID of the object
Visible	Boolean	Read	Visibility of the marker
Transformation	Matrix	Read	Transformation of the marker from the camera
Position	Vector	Read	Position of the marker from the camera (in millimeters)
Orientation	Vector	Read	Orientation of the marker form the camera in Euler angle (in degrees)

vobject

For declaring a virtual object.

<vobject name="*object-name*" model="*model-file-name*">
property-list
</vobject>

object-name name of the object in string
model-file-name file name of the model file
property-list a list of property tags

(ex)

```
<vobject name="fish" model="Models/fish.obj">
    <property name="BaseObject" value="page01"/>
    <property name="Position" value="10 20 30.5"/>
    <property name="Orientation" value="10 0.0 120"/>
    <property name="Scale" value="1.5"/>
</vobject>
```

Properties of vobject

Name	Data type	Read/Write	Description
ID	Scalar	Read	ID of the object
Visible	Boolean	Read/Write	Visibility of the object
BaseObject	Scalar	Read/Write	ID of the parent object. (use object names instead of ID numbers in TARML)
Transformation	Matrix	Read/Write	Transformation of the object from the camera
Position	Vector	Read/Write	Position of the object relative to its parent object (in millimeters)
Orientation	Vector	Read/Write	Orientation of the object relative to its parent object in Euler angle (in degrees)
Scale	Scalar	Read/Write	Scale factor
Color	Color	Read/Write	RGB values in triplet decimal numbers. Each component ranges from 0.0 to 1.0 (This property is deprecated.)

sobject

For declaring a sound object

<sobject name="*object-name*" sample="*wave-file-name*">
property-list
</sobject>

object-name name of the object in string
model-file-name file name of the sound file
property-list a list of property tags

(ex)

```
<sobject name="ambientSound" sample="Sounds/birds-stream.wav">
    <property name="Play" value="1"/>
    <property name="Loop" value="1"/>
    <property name="BaseObject" value="card00"/>
</sobject>
```

Properties of sobject

Name	Data type	Read/Write	Description
ID	Scalar	Read	ID of the object
Play	Boolean	Read/Write	Playing status
Loop	Boolean	Read/Write	Whether to loop the sound
BaseObject	Scalar	Read/Write	ID of the parent object. (use object names instead of ID numbers in TARML)
Transformation	Matrix	Read/Write	Transformation of the object from the camera
Position	Vector	Read/Write	Position of the object relative to its parent object (in millimeters)
Orientation	Vector	Read/Write	Orientation of the object relative to its parent object in Euler angle (in degrees)

lobject

For declaring logical entities.

<lobject name="*object-name*" logic="*logic-type*">
property-list
</lobject>

object-name name of the object in string
logic-type type of the logic to use
(default types) AND, OR, NOT, Distance, MotorX, MotorY, MotorZ
property-list a list of property tags

(ex)

```
<lobject name="motor" logic="MotorX">
        <property name="BaseObject" value="page01"/>
        <property name="RPM" value="10"/>
        <property name="On" value="1"/>
</lobject>
```

Common properties of lobject

Name	Data type	Read/Write	Description
ID	Scalar	Read	ID of the object
BaseObject	Scalar	Read/Write	ID of the parent object. (use object names instead of ID numbers in TARML)
Transformation	Matrix	Read/Write	Transformation of the object from the camera
Position	Vector	Read/Write	Position of the object relative to its parent object (in millimeters)
Orientation	Vector	Read/Write	Orientation of the object relative to its parent object in Euler angle (in degrees)

Additional properties of AND, OR type logic

Name	Data type	Read/Write	Description
Input1	Boolean	Write	Input 1
Input2	Boolean	Write	Input 2
Output	Boolean	Read	Ouput value AND: Input1 & Input2 OR: Input1 | Input2

Additional properties of NOT logic type

Name	Data type	Read/Write	Description
Input	Boolean	Write	Input
Output	Boolean	Read	Ouput value (~Input)

Additional properties of Distance type logic

Name	Data type	Read/Write	Description
Position1	Vector	Write	Position to compare
Position2	Vector	Write	Position to compare
Threshold	Scalar	Write	Distance threshold to determine near or far
Near	Boolean	Read	1 if |Position1-Position2| < Threshold 0 if |Position1-Position2| >= Threshold
Far	Boolean	Read	0 if |Position1-Position2| < Threshold 1 if |Position1-Position2| >= Threshold

Additional properties of MotorX, MotorY and MotorZ type logic

Name	Data type	Read/Write	Description
RPM	Scalar	Write	Round per minute
On	Booelan	Write	1 to turn on, 0 to turn off
Motor	Vector	Read	Orientation of the motor in Euler angle (in degrees)

link

link between two properties

<link source="*src-object*:*src-property*" target="*tgt-object*:*tgt-property*"/>

src-object source object name
src-property source property name
tgt-object target object name
tgt-property target property name

(ex)

```
<link source="card01:Visible" target="ambientSound:Play"/>
```

Sample application code

```
<tarml>
	<pobject name="card01" marker="Markers/card-40.patt">
	</pobject>
	<vobject name="fish" model="Models/fish.obj">
		<property name="BaseObject" value="card01"/>
		<property name="Position" value="10 20 30.5"/>
	</vobject>
	<lobject name="motor" logic="MotorZ">
		<property name="RPM" value="30"/>
	</lobject>
	<link source="motor:ShaftOrientation" target="fish:Orientation"/>
	<sobject name="bubbleSound" sample="Sounds/bubble.wav">
	</sobject>
	<link source="card01:Visible" target="bubbleSound:Play"/>
</tarml>
```

Appendix B. ANOVA Results

B.1. Total task completion time

The ANOVA Procedure

Class Level Information

Class	Levels	Values
SUBJ	12	1 2 3 4 5 6 7 8 9 10 11 12
IA	2	0 1

Number of observations 24

Dependent Variable: TOTTIME tottime */* *in seconds* */*

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	23	5990413.333	260452.754	.	.
Error	0	0.000	.		
Corrected Total	23	5990413.333			

R-Square	Coeff Var	Root MSE	TOTTIME Mean
1.000000	.	.	1989.667

Source	DF	Anova SS	Mean Square	F Value	Pr > F
SUBJ	11	2666938.333	242448.939	.	.
IA	1	2564988.167	2564988.167	.	.
SUBJ*IA	11	758486.833	68953.348	.	.

```
Tests of Hypotheses Using the Anova MS for SUBJ*IA as an Error Term
Source              DF         Anova SS      Mean Square    F Value    Pr > F
IA                   1      2564988.167      2564988.167      37.20    <.0001

Level of            ----------TOTTIME----------
IA             N            Mean          Std Dev
0             12       2316.58333       464.235727
1             12       1662.75000       309.657032
```

B.2. Total task completion time in task types

```
/* spatial tasks */
The ANOVA Procedure

Class Level Information
Class    Levels    Values
SUBJ     12        1 2 3 4 5 6 7 8 9 10 11 12
IA        2        0 1
Number of observations    24

Dependent Variable: TOTSTIME   totstime   /* in seconds */

Source           DF   Sum of Squares     Mean Square    F Value    Pr > F
Model            23     4699432.000      204323.130         .         .
Error             0           0.000               .
Corrected Total  23     4699432.000
```

```
R-Square     Coeff Var      Root MSE    TOTSTIME Mean
1.000000           .              .         1131.500

Source  DF        Anova SS     Mean Square    F Value    Pr > F
SUBJ    11      838875.000       76261.364          .         .
IA       1     3477770.667     3477770.667          .         .
SUBJ*IA 11      382786.333       34798.758          .         .

Tests of Hypotheses Using the Anova MS for SUBJ*IA as an Error Term

Source DF        Anova SS     Mean Square    F Value    Pr > F
IA      1     3477770.667     3477770.667      99.94    <.0001

Level of            ----------TOTSTIME----------
IA             N             Mean          Std Dev
0             12       1512.16667       286.538537
1             12        750.83333       170.164003

/* non-spatial tasks */
The ANOVA Procedure

Class Level Information
Class    Levels    Values
SUBJ         12    1 2 3 4 5 6 7 8 9 10 11 12
IA            2    0 1
Number of observations    24
```

```
Dependent Variable: TOTLTIME   totltime   /* in seconds */

Source          DF   Sum of Squares     Mean Square    F Value    Pr > F
Model           23     815477.3333      35455.5362         .         .
Error            0          0.0000            .
Corrected Total 23     815477.3333

R-Square     Coeff Var      Root MSE    TOTLTIME Mean
1.000000           .              .         858.1667

Source  DF        Anova SS     Mean Square    F Value    Pr > F
SUBJ    11     623096.3333      56645.1212         .         .
IA       1      69337.5000      69337.5000         .         .
SUBJ*IA 11     123043.5000      11185.7727         .         .

Tests of Hypotheses Using the Anova MS for SUBJ*IA as an Error Term
Source  DF        Anova SS     Mean Square    F Value    Pr > F
IA       1     69337.50000     69337.50000        6.20    0.0301

Level of            ----------TOTLTIME----------
IA            N             Mean          Std Dev
0            12       804.416667       207.915393
1            12       911.916667       156.850513
```

B.3. Average task completion time between authoring tools

```
------------------------ spatial=0 --------------------------------
The ANOVA Procedure

Class Level Information
Class         Levels    Values
SUBJ              12    1 2 3 4 5 6 7 8 9 10 11 12
IA                 2    0 1
VR                 2    0 1
Number of observations    72

Dependent Variable: TIME   time   /* in seconds */

Source           DF  Sum of Squares     Mean Square    F Value    Pr > F
Model            23     271825.7778      11818.5121       1.92    0.0288
Error            48     295996.0000       6166.5833
Corrected Total  71     567821.7778

R-Square     Coeff Var      Root MSE     TIME Mean
0.478717      27.45187      78.52760      286.0556

Source       DF        Anova SS     Mean Square    F Value    Pr > F
SUBJ(VR)     10     192791.8889      19279.1889       3.13    0.0038
IA            1      23112.5000      23112.5000       3.75    0.0588
SUBJ*IA(VR)  10      15551.7778       1555.1778       0.25    0.9884
VR            1      14906.8889      14906.8889       2.42    0.1266
IA*VR         1      25462.7222      25462.7222       4.13    0.0477
```

Tests of Hypotheses Using the Anova MS for SUBJ*IA(VR) as an Error Term

Source	DF	Anova SS	Mean Square	F Value	Pr > F
IA	1	23112.50000	23112.50000	14.86	0.0032
IA*VR	1	25462.72222	25462.72222	16.37	0.0023

Tests of Hypotheses Using the Anova MS for SUBJ(VR) as an Error Term

Source	DF	Anova SS	Mean Square	F Value	Pr > F
VR	1	14906.88889	14906.88889	0.77	0.3999

Level of	TIME		
IA	N	Mean	Std Dev
0	36	268.138889	93.8342468
1	36	303.972222	82.2086148

Level of	TIME		
VR	N	Mean	Std Dev
0	36	300.444444	87.0807325
1	36	271.666667	90.6339577

Level of	Level of	TIME		
IA	VR	N	Mean	Std Dev
0	0	18	301.333333	94.8962282
0	1	18	234.944444	82.3953684
1	0	18	299.555556	81.2723976
1	1	18	308.388889	85.2499018

```
------------------------ spatial=1 --------------------------------
The ANOVA Procedure

Class Level Information
Class         Levels    Values
SUBJ              12    1 2 3 4 5 6 7 8 9 10 11 12
IA                 2    0 1
VR                 2    0 1
Number of observations    72

Dependent Variable: TIME   time   /* in seconds */

Source           DF  Sum of Squares     Mean Square    F Value    Pr > F
Model            23     1566477.333       68107.710      11.24    <.0001
Error            48      290958.667        6061.639
Corrected Total  71     1857436.000

R-Square     Coeff Var      Root MSE     TIME Mean
0.843355      20.64247      77.85653      377.1667

Source       DF        Anova SS     Mean Square    F Value    Pr > F
SUBJ(VR)     10      246084.500       24608.450       4.06    0.0004
IA            1     1159256.889     1159256.889     191.24    <.0001
SUBJ*IA(VR)  10       88302.056        8830.206       1.46    0.1853
VR            1       33540.500       33540.500       5.53    0.0228
IA*VR         1       39293.389       39293.389       6.48    0.0142
```

Tests of Hypotheses Using the Anova MS for SUBJ*IA(VR) as an Error Term

Source	DF	Anova SS	Mean Square	F Value	Pr > F
IA	1	1159256.889	1159256.889	131.28	<.0001
IA*VR	1	39293.389	39293.389	4.45	0.0611

Tests of Hypotheses Using the Anova MS for SUBJ(VR) as an Error Term

Source	DF	Anova SS	Mean Square	F Value	Pr > F
VR	1	33540.50000	33540.50000	1.36	0.2701

Level of	TIME		
IA	N	Mean	Std Dev
0	36	504.055556	120.028555
1	36	250.277778	74.438704

Level of	TIME		
VR	N	Mean	Std Dev
0	36	398.750000	178.677903
1	36	355.583333	142.075709

Level of IA	Level of VR	TIME N	TIME Mean	TIME Std Dev
0	0	18	549.000000	101.599560
0	1	18	459.111111	122.723452
1	0	18	248.500000	87.184220
1	1	18	252.055556	61.647453

B.4. Subjective ratings

```
/* easy to learn */
The ANOVA Procedure

Class Level Information
Class         Levels    Values
SUBJ              12    1 2 3 4 5 6 7 8 9 10 11 12
IA                 2    0 1
Number of observations    24

Dependent Variable: EASYLEARN   easylearn   /* rating in 7-point scale */
Source           DF  Sum of Squares     Mean Square    F Value    Pr > F
Model            23     33.95833333      1.47644928          .        .
Error             0      0.00000000       .
Corrected Total  23     33.95833333

R-Square     Coeff Var      Root MSE    EASYLEARN Mean
1.000000           .              .          4.208333

Source   DF        Anova SS     Mean Square    F Value    Pr > F
SUBJ     11     26.45833333      2.40530303          .        .
IA        1      2.04166667      2.04166667          .        .
SUBJ*IA  11      5.45833333      0.49621212          .        .

Tests of Hypotheses Using the Anova MS for SUBJ*IA as an Error Term
Source   DF        Anova SS     Mean Square    F Value    Pr > F
IA        1      2.04166667      2.04166667       4.11    0.0674
```

```
Level of            ----------EASYLEARN----------
IA             N              Mean          Std Dev
0             12       3.91666667       1.24011241
1             12       4.50000000       1.16774842

/* easy to use */
The ANOVA Procedure
Class Level Information
Class         Levels    Values
SUBJ              12    1 2 3 4 5 6 7 8 9 10 11 12
IA                 2    0 1
Number of observations    24

Dependent Variable: EASYUSE   easyuse   /* rating in 7-point scale */
Source           DF  Sum of Squares     Mean Square    F Value    Pr > F
Model            23     25.33333333      1.10144928          .         .
Error             0      0.00000000       .
Corrected Total  23     25.33333333

R-Square     Coeff Var      Root MSE    EASYUSE Mean
1.000000           .              .         4.166667

Source     DF        Anova SS     Mean Square    F Value    Pr > F
SUBJ       11     14.33333333      1.30303030          .         .
IA          1      0.16666667      0.16666667          .         .
SUBJ*IA    11     10.83333333      0.98484848          .         .
```

```
Tests of Hypotheses Using the Anova MS for SUBJ*IA as an Error Term
Source    DF        Anova SS     Mean Square    F Value    Pr > F
IA         1      0.16666667      0.16666667       0.17    0.6887

Level of            ----------EASYUSE----------
IA            N             Mean          Std Dev
0            12       4.25000000       1.13818037
1            12       4.08333333       0.99620492

/* confidence */
The ANOVA Procedure

Class Level Information
Class         Levels    Values
SUBJ              12    1 2 3 4 5 6 7 8 9 10 11 12
IA                 2    0 1
Number of observations    24

Dependent Variable: CONF   conf   /* rating in 7-point scale */
Source           DF    Sum of Squares     Mean Square    F Value    Pr > F
Model            23     25.95833333      1.12862319          .          .
Error             0      0.00000000       .
Corrected Total  23     25.95833333

R-Square     Coeff Var      Root MSE     CONF Mean
1.000000           .              .      4.208333
```

Source	DF	Anova SS	Mean Square	F Value	Pr > F
SUBJ	11	16.45833333	1.49621212	.	.
IA	1	2.04166667	2.04166667	.	.
SUBJ*IA	11	7.45833333	0.67803030	.	.

Tests of Hypotheses Using the Anova MS for SUBJ*IA as an Error Term

Source	DF	Anova SS	Mean Square	F Value	Pr > F
IA	1	2.04166667	2.04166667	3.01	0.1106

Level of	CONF		
IA	N	Mean	Std Dev
0	12	4.50000000	0.90453403
1	12	3.91666667	1.16450015

References

[1] M. F. Aburdene, *Computer Simulation of Dynamic Systems*, Wm. C. Brown Publishers, 1988.

[2] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.

[3] M. Billinghurst, I. Poupyrev, H. Kato and R. May, "Mixing Realities in Shared Space: An Augmented Reality Interface for Collaborative Computing," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2000)*, New York, U.S.A., July 30-August 2, 2000, pp. 1641-1644.

[4] M. Billinghusrt, H. Kato, I. Poupyrev, "The MagicBook – Moving Seamlessly between Reality and Virtuality," in *IEEE Computer Graphics and Applications*, Vol.21, No.3, 2001, pp.6-8.

[5] D. A. Bowman and L. F. Hodges, "User Interface Constraints for Immersive Virtual Environment Applications," *Technical Report of Graphics, Visualization, and Usability Center*, Georgia Institute of Technology, GIT-GVU-95-26, 1995.

[6] J. Butterworth, A. Davidson, S. Hench and T. M. Olano, "3DM: A Three Dimensional Modeler Using a Head-Mounted Display," in *Proceedings of Symposium on Interactive 3D Graphics (I3D '92)*, Cambridge, MA, U.S.A., 1992, pp.135-138.

[7] M. Callaghan, H. Hirschmuller, "3-D Visualization of Design Patterns and Java Programs in Computer Science Education," in *Proceedings of Annual Joint Conference of Integrating Technology into Computer Science Education '98*, Dublin, Ireland, August 18-21, 1998, pp.37-40.

[8] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice*, 2nd ed., Addison-Wesley, 1990.

[9] D. Harel, H. Lachover, A. Naamad, A. Pnuell, M. Politi, R. Sherman, A. Shtull-Trauring and M. Trakhtenbrot, "STATEMATE: A Working Environment for the Development of Complex Reactive Systems," in *IEEE Transactions on Software Engineering*, Vol.16, No.4, 1990, pp. 403-414.

[10] C. Heeter, "Being There: The subjective experience of presence, telepresence," in *Presence: Teleoperators and Virtual Environments*, Vol.1, No.2, MIT Press, 1992, pp. 262-271.

[11] R. Holm, E. Stauder, R. Wagner, M. Priglinger and J. Volkert, "A Combined Immersive and Desktop Authoring Tool for Virtual Environments," in *Proceedings of the IEEE Virtual Reality 2002*, Orlando, FL, U.S.A., March 24-28, 2002, pp. 93-100.

[12] R. S. Kalawsky, *The Science of Virtual Reality and Virtual Environments*, Addison-Wesley, 1993, pp. 212-219.

[13] H. Kato, M. Billinghurst, I. Poupyrev, N. Tetsutani and K. Tachibana, "Tangible Augmented Reality for Human Computer Interaction," in *Proceedings of Nicograph 2001*, Nagoya, Japan, 2001.

[14] G. J. Kim, K. Kang, H. Kim and J. Lee, "Software Engineering of Virtual Worlds," in *Proceedings of Virtual Reality Software & Technology '98*, Taipei, Taiwan, November 2-5, 1998, pp.131-139.

[15] S. Kim and G. J. Kim, "Using Keyboards with Head Mounted Displays," in

Proceedings of ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI) 2004, NTU, Singapore, June 16-18, 2004, pp. 336-343.

[16] G. A. Lee, M. Billinghurst and G. J. Kim, "Occlusion based Interaction Methods for Tangible Augmented Reality Environments," in *Proceedings of ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI) 2004*, NTU, Singapore, June 16-18, 2004, pp. 419-426.

[17] G. A. Lee, C. Nelles, M. Billinghurst and G. J. Kim, "Immersive Authoring of Tangible Augmented Reality Applications," in *Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2004)*, Arlington, VA, U.S.A., November 2-5, 2004, pp.172-181.

[18] J. Liang and M. Green, "JDCAD: A Highly Interactive 3D Modeling System," in *Computer & Graphics*, Vol.18, No.4, 1994, pp.499-506.

[19] H. Lieberman (ed.), *Your wish is my command: Programming by Example*, Morgan Kaufmann Publishers, 2001.

[20] M. R. Mine, "ISAAC: A Virtual Environment Tool for the Interactive Construction of Virtual Worlds," *Technical Report of Department of Computer Science, UNC Chapel Hill, CS TR95-020*, 1995.

[21] M. A. Najork, "Programming in Three Dimensions," in *Journal of Visual Languages and Computing*, Vol.7, No.2, 1996, pp.219-242.

[22] J. T. Nosek, "The Case for Collaborative Programming," in *Communications of the*

ACM, Vol.41 No.3, 1998, pp.105-108.

[23] M. Rabiger, *Directing: Film Techniques and Aesthetics*, Focal Press, 1997.

[24] J. Seo and G. J. Kim, "A Structured approach to Virtual Reality System Design," in *Presence: Teleoperators and Virtual Environments*, Vol.11 No.4, MIT Press, August, 2002, pp.378-403.

[25] J. Seo, *A Structured Methodology for Virtual Reality Systems Development and Interactive Support Tools*, Ph.D. Dissertation, Department of Computer Science and Engineering, POSTECH, 2005.

[26] B. Shneiderman, Foreword in Henry Lieberman(ed.), *Your wish is my command: Programming by Example*, Morgan Kaufmann Publishers, 2001, pp. v-vii.

[27] M. Slater and M. Usoh, "Representations Systems, Perceptual Position, and Presence in Immersive Virtual Environments," in *Presence: Teleoperators and Virtual Environments*, Vol.2, No.3, MIT Press, 1993, pp.221-233.

[28] M. Slater, M. Usoh and A. Steed, "Taking Steps: The Influence of Walking Technique on Presence in Virtual Reality," in *ACM Transactions on Computer-Human Interaction*, Vol.2, No.3, 1995, pp.201-219.

[29] D. C. Smith, "KIDSIM: Programming Agents without a Programming Language," in *Communications of the ACM*, Vol.37 No.7, 1994, pp.55-67.

[30] C. Soanes (ed.), *Compact Oxford English Dictionary of Current English* (2nd ed.), Oxford University Press, 2003.

[31] A. Steed and M. Slater, “A Dataflow Representation for Defining Behaviours within Virtual Environments,” in *Proceedings of Virtual Reality Annual International Symposium ’96*, March 30 - April 3, 1996, pp.163-167.

[32] J. Steuer, “Defining Virtual Reality: Dimensions Determining Telepresence,” in *The Journal of Communication*, Vol.42, No.4, 1992, pp.73-93.

[33] R. Stiles and M. Pontecorvo, “Lingua Graphica : A Visual Language for Virtual Environments,” in *IEEE Workshop on Visual Languages*, IEEE Computer Society Press, September 15-18, 1992, pp.225-227.

[34] R. Stoakley, M. J. Conway and R. Pausch, “Virtual Reality on WIM: Interactive Worlds in Miniature,” in *Proceedings of International Conference for Human-Computer Interaction (SIGCHI’95)*, 1995, pp. 265-272.

[35] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, 1998.

[36] G. Wesche and H. Seidel, “FreeDrawer – A Free-Form Sketching System on the Responsive Workbench,” in *Proceedings of Virtual Reality Software & Technology ’01*, Alberta, Canada, November 15-17, 2001, pp.167-174.

[37] L. Williams and R. Kessler, “All I really need to know about pair programming I learned in kindergarten,” in *Communications of the ACM*, Vol.43, No.5, 2000, pp.108-114.

[38] D. Wolber, “Pavlov: An Interface Builder for Designing Animated Interfaces,” in *ACM Transactions on Computer Human Interaction*, Vol.4, No.4, 1997, p.347-386.

[39] 3D GameStudio. http://www.3dgamestudio.com

[40] 5DT Data glove. http://www.5dt.com

[41] Alice. http://www.alice.org

[42] ARToolKit. http://www.hitl.washington.edu/artoolkit

[43] Crystal Space. http://www.crystalspace3d.org

[44] Extreme Programming. http://www.extremeprogramming.org

[45] Fastrak, Polhemus. http://www.polhemus.com

[46] Lingo Language. http://www.adobe.com/support/director/lingo.html

[47] Nebula Device, Radon Labs. http://www.radonlabs.de/nebula.html

[48] Performer. http://www.sgi.com/software/performer

[49] Python Language. http://www.python.org

[50] Virtual Reality Markup Language. http://www.web3d.org

[51] VR Juggler. http://www.vrjuggler.org

[52] World Tool Kit. http://www.sense8.com

요약문

디지털 콘텐트(digital content)는 지난 30 여 년간 많은 발전을 거쳐왔다. 초기에는 단순한 텍스트와 영상으로 구성되던 디지털 콘텐트가, 이제는 동영상과 멀티미디어의 형태를 띠고 있으며, 나아가 일반인들이 직접 콘텐트를 만들고 이를 인터넷을 통해 공유하는 등, 디지털 콘텐트는 실로 우리 생활 가까이 다가와 있다. 이러한 멀티미디어 콘텐트의 보편화는 사용하기 쉽고 편리한 저작 기술의 발전이 뒷받침 되었기에 가능했다고 해도 과언이 아닐 것이다. 반면, 가상현실(virtual reality) 콘텐트의 개발 과정은 아직도 프로그래밍 작업에 의존하고 있어 프로그램 개발자 외의 사람들이 사용하기에는 불편한 것이 현실이다.

프로듀서, 감독 등의 예술적인 배경을 갖는 사람들, 나아가 일반인들이 쉽고 편리하게 사용할 수 있는 가상현실 콘텐트 저작도구를 실현하기 위해, 본 논문에서는 저작 공간과 실행 공간이 일치되는, 즉, 가상환경 내에서 가상세계(virtual world)를 저작하는, 이른바 몰입형 저작 기법(immersive authoring method)을 제안한다. 몰입형 저작 기법의 가장 큰 특징은 저작 환경 자체가 몰입형 공간이므로, 저작자가 최종 결과물을 직접 경험하면서, 3 차원 상호작용 기법들을 사용하여 쉽고 편리하게 가상현실 콘텐트를 저작 할 수 있는 점이다. 본 논문에서는 몰입형 저작 기법의 기본 개념과 원리를 제시하고, 가상현실 콘텐트 저작 작업의 분석을 바탕으로 저작도구에 몰입형 저작 기법을 적용하기 위한 방안을 제안하며, 몰입형 저작도구의 일반적인 구조와 개발 과정을 제시함으로써 몰입형 저작 기법의 구현 가능성을 보인다. 또한, 이를 바탕으로, 본 연구에서 개발된 몰입형 저작도구인 PiP 시스템과 iaTAR 시스템과 이들을 이용한 콘텐트 저작 사례들을 살펴봄으로써 몰입형 저작 기법의 유용성을 고찰한다.

몰입형 저작 기법의 효율성 및 효용성에 대한 객관적인 검증을 위해 본 연구에서는 사용자 실험을 수행하였다. 본 실험에서는 몰입형 저작 기법과 기존의 저작 방법을 사용하여 동일한 콘텐트를 저작할 때의 효율성 및 사용성을 비교 평가 하였다. 몰입형 저작도구로는 iaTAR 을, 기존의 저작 방법으로는 TARML 스크립트 언어를 사용하였으며, 피실험자는 각각의 도구를 사용하여 제시된 사양의 콘텐트를 저작하였다. 실험 결과, 제시된 콘텐트를 저작하는데 걸린 시간을 비교 분석함으로써 몰입형 저작 기법이 기존의 방법에 비해 효율적이라는 결론을 얻었다. 특히, 공간적인 저작 작업의 경우 몰입형 저작 기법의 효율성이 두 배에 달하는 것으로 나타나, 몰입형 저작 기법의 효율성에 대해 매우 긍정적인 결론을 내릴 수 있었다. 몰입형 저작도구의 사용성과 편리성에 대해서는, 비록 설문결과의 분석을 통해서는 기존의 방법과 비교해 유의한 차이가 없는 것으로 나타났으나, 실험 중에 피실험자가 저작도구의 사용설명서를 참조하거나 사용 방법을 잊은 회수를 비교해 볼 때, 몰입형 저작 기법이 사용하고 배우기에 비교적 쉬움을 알 수 있었다. 또한, 설문을 통해 몰입형 저작 기법은 가상객체의 배치와 같은 공간적인 작업에 유용하며, 기존의 방법은 논리적인 표현과 같은 비공간적인 작업에 유용하다는 점을 확인 할 수 있었다. 이러한 점들을 종합해 볼 때, 몰입형 저작 도구에서도 2 차원 인터페이스를 효율적으로 사용할 수 있도록 개선 한다면, 두 가지 저작 방법의 장점을 통합하는 형태의 이상적인 저작 환경의 실현이 가능할 것으로 생각된다.

아직까지 가상현실 장비들의 기능상 한계로 장시간 사용하는데 불편함도 있지만, 이를 개선하고 다양한 형태의 가상현실 콘텐트를 표현할 수 있는 가상세계 모델이 정립된다면, 앞으로 가상현실 콘텐트를 효율적으로 저작하는데 몰입형 저작 기법이 많은 도움을 줄 것으로 전망된다.

감사의 글

"여호와의 말씀에 내 생각은 너희 생각과 다르며 내 길은 너희 길과 달라서 하늘이 땅보다 높음 같이 내 길은 너희 길보다 높으며 내 생각은 너희 생각보다 높으니라" – 이사야 55 장 8~9 절

에벤에셀, 할렐루야!

처음에 제가 생각했던 것과는 많이 다른 길로 여기까지 왔습니다. 하지만, 분명한 것은 여호와 하나님께서는 항상 저의 계획보다 더 좋은 계획을 가지고 계셨고, 그 길로 나를 인도 하셨으며, 또 앞으로도 변함이 없으실 것이라는 점입니다. 앞으로도 항상 그분의 뜻에 순종하는 법을 배워가기를 원합니다.

이 논문이 나오기까지 저를 잘 지도해주시고 아낌없는 도움을 베풀어 주신 박찬모 교수님, 김정현 교수님, 최승문 교수님, 그리고 강교철 교수님, 한성호 교수님, 정순기 교수님께 깊은 감사의 말씀을 올립니다. 연구뿐 아니라, 교수님들의 모습 속에서 많은 것들을 배웁니다. 감사합니다!

가상현실이라는 아직은 좁은 길을 함께 걸어 온 포항공대 가상현실 연구실의 여러 선후배님들, 그리고 한국전자통신연구원 콘텐츠연구본부, 가상현실연구팀 원들께도 감사의 마음과 격려를 보냅니다.

끝으로, 항상 기도로 저를 격려해 주시는 부모님, 예쁜 딸을 키우느라 고생하는 아내와 다른 모든 가족들에게 감사와 사랑의 마음을 전합니다.

이 력 서

성　　명 : 이 건 (李 鍵, Gun A. Lee)

생년월일 : 1977 년 11 월 22 일

출 생 지 : 서울특별시

주　　소 : 경북 성주군 선남면 명포리 712

e-mail : endovert@postech.ac.kr

학　　력

1996.3 ~ 2000.2 : 경북대학교 자연과학대학 컴퓨터과학과 학사 (B.S.)

2000.3 ~ 2002.2 : 포항공과대학교 대학원 컴퓨터공학과 석사 (M.S.)

2002.3 ~ 2009.2 : 포항공과대학교 대학원 컴퓨터공학과 박사 (Ph.D.)

발 표 논 문

< Book Chapter >

Gun A. Lee, Gerard J. Kim and Mark Billinghurst, “Chapter XIII. Interaction Design for Tangible Augmented Reality Applications,” in *Emerging Technologies of Augmented Reality: Interfaces and Design* (Michael Haller, Mark Billinghurst and Bruce Thomas ed.), IDEA Group Publishing, 2006, pp.261-281.

< International Journal >

Gun A. Lee and Gerard J. Kim, “Immersive authoring of Tangible Augmented Reality content: A user study,” in *Journal of Visual Languages & Computing*, In press, January 2009.

Gun A. Lee, Gerard J. Kim and Mark Billinghurst, “Immersive Authoring: What You eXperience Is What You Get,” in *Communications of the ACM 48* (7), July 2005, pp.76-81.

< International Conference >

Gun A. Lee, Gerard Jounghyun Kim, Mark Billinghurst, “Directing Virtual Worlds: Authoring and Testing for/within Virtual Reality based Contents,” in *Proceedings of the 14th International Conference on Artificial Reality and Telexistence (ICAT)*, Seoul, Korea, Nov.30-Dec.2, 2004, pp.534-537.

Gun A. Lee, Claudia Nelles, Mark Billinghurst and Gerard Jounghyun Kim, “Immersive Authoring of Tangible Augmented Reality Applications,” in *Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, Arlington, VA, U.S.A., Nov.2-5, 2004, pp.172-181.

Gun A. Lee, Mark Billinghurst and Gerard Jounghyun Kim, “Occlusion based Interaction Methods for Tangible Augmented Reality Environments,” in *Proceedings of ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI)*, NTU, Singapore, Jun.16-18, 2004, pp.419-426.

Gun A. Lee, Gerard Jounghyun Kim and Chan-Mo Park, “Modeling Virtual Object Behavior within Virtual Environment,” in *Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST)*, Hong Kong, China, Nov.11-13, 2002, pp.41-48.

< Korean Journal >

이 건, 서진석, 김정현, “컴포넌트 기반 프레임워크를 이용한 가상객체의 재사용 기법 (Reusing Virtual Objects in a Component based Framework)”, *디지털엔터테인먼*

트 학회 논문지, 제 1 권 1 호, 2007 년 12 월 발행, pp.33-38.

김정현, 이 건, "디지털 컨텐트의 몰입형 저작 기법: 나도 감독이 되어보자", *Graphics Live*, 2004 년 10 월호 (vol.54), pp.91-94.

< Korean Conference >

이 건, 김정현, 마크 빌링허스트, "가상현실 기반 컨텐트의 몰입형 저작 기법", *한국 HCI 학술대회 발표논문집*, 대구, 2005 년 1 월 31 일-2 월 3 일.

김태경, 한성호, 김광재, 김정현, 김종서, 홍상우, 이자연, 이 건, 임정훈, "3 차원 프로토타입을 활용한 최적 감성 설계 지원 시스템 (Affective Product Design System using 3D prototype)", *대한인간공학회 학술대회논문집*, 강원도 용평, 2002 년 11 월 28-29 일, pp.33-36.

양웅연, 이 건, 이자연, 복일근, 한성호, "3 차원 다관절 물체의 형태를 이해하기 위한 관찰 방향을 선택하는 방법의 설계 및 실험", *한국 HCI 학술대회 발표논문집*, 강원도 평창, 2002 년 2 월 4-7 일.

신선형, 조동식, 조창석, 이 건, 김남규, 김정현, 박찬모, 추성열, "대형 화면 기반의 데이터 시각화를 위한 상호작용", *한국 HCI 학술대회 발표논문집*, 강원도 평창, 2002 년 2 월 4-7 일.