In Virtual Realities Dagstuhl Seminar 20013, 2015 (to appear). This is the manuscript of the authors.

Content Creation and Authoring Challenges for Virtual Environments: from User Interfaces to Autonomous Virtual Characters

Ralf Dörner[1], Marcelo Kallmann[2], and Yazhou Huang[2]

[1] RheinMain University of Applied Sciences, Wiesbaden, Germany
[2] University of California, Merced, USA

Abstract. How is content for virtual environments (VEs) created? How is behavior and motion in VEs specified? How are user interfaces designed and implemented for VEs? Authoring is a crucial aspect for using VEs successfully. This chapter addresses some of the current challenges in authoring VEs and recent research directions that are being explored in order to address these challenges. One highly relevant use case is the definition of motions to be executed by virtual characters. For this, motion modeling interfaces are presented that facilitate the process of programming the motions of virtual agents in VEs.

Keywords: Authoring Process, Authoring Tools, Virtual Environments, Motion Modeling, Virtual Characters, User Interfaces for Virtual Environments

1 Introduction

Since the first virtual environment has been created in the 1960s [61] there have been tremendous advances not only in the basic hardware and software but also in the quality and complexity of the content. While the first VEs were static, most VEs now contain animated scenes and autonomous entities. Whereas initially the specification of animation was mostly based on methods such as user-triggered keyframe animations, now the definition of animations has evolved to an abstract behavioral level e.g. by describing overall goals. This leaves the execution of low-level behavior (e.g. path finding, obstacle avoidance, object interaction) to sophisticated methods from artificial intelligence that are capable of generating complex emergent behavior. Today VEs are expected to contain not only 3D models but highly complex entities such as autonomous virtual characters who behave autonomously in complex scenarios. This evolution has led to exciting new applications in strategy games, simulation-based training, and synthetically populated scenes in movies. This chapter addresses a crucial issue that is directly affected by the rising complexity: *authoring* of virtual environments.

A Virtual Environment (VE) allows users to experience a virtual world and to feel present in it. Here, the term *virtual world* denotes the content of a VE. *Content* consists of geometric 3D models of entities, their position and orientation in 3D space, other media (e.g. audio) associated with them, descriptions of their appearance and physical properties, descriptions of their behavior (e.g. in the form of simulation models) and specifications how users can interact with individual entities or the whole virtual world (e.g. navigation, selection, manipulation). Authors create the content. For instance, in a Virtual Reality training application for critical decision making such as [52] where agents engage the learner in a game-like scenario, the authors needed to create a 3D model of a virtual airport, create and animate 3D models of the agents, specify the behavior of the agents and specify the simulation of the virtual world. Usually, there are several authors involved which are required to possess different expertise. While one author needs the ability to animate a virtual human, another author might need knowledge in the application domain, e.g. to specify a training scenario. Authoring of the content is only one aspect of creating a working VE. For example, one integral part is the implementation of the underlying VR systems (e.g. the realization of image rendering, collision detection or physics simulation).

Another integral part of VEs is the user interface. With the advances in hardware, software and animated content of VEs, their *user interfaces (UIs)* also became more complex and sophisticated. Every VE has to include an interface for interaction between the user and the virtual world presented by the underlying computer system. Hence, VEs can be considered as a specific research subject and application field of human computer interaction (HCI). The HCI methodologies used for VEs differ considerably from graphical UIs that often rely on the WIMP (windows, icons, menus, pointer)-metaphor. The term *post-WIMP UIs* has been coined for UIs that make use of advanced technology, e.g. tracking technology [13]. They equip real time interactive systems with a plethora of interaction techniques stemming not exclusively from virtual reality and augmented reality, but also from gesture-based UIs, speech-based UIs, multi-touch and tangible user interfaces (i.e. a user interface where real objects called props are used as proxies for interaction). Post-WIMP UIs are becoming main stream with the advent of reasonably prized hardware such as digital cameras in smart phones and smart tablets, the Oculus Rift head mounted display [25], or the Leap Motion [24] or Microsoft Kinect [47] depth sensor. This is a novel situation for UIs that are employed by VEs which traditionally have been used only by small user groups. With more people involved and commercial products available, the developments in this area have become highly dynamic. As a result, UIs for VEs are rapidly exploring new paradigms of interaction that are increasingly more difficult to author. This also exceeds the scope of traditional 3D user interfaces [7] that have been employed in VEs.

Why is authoring important for VEs? If authoring processes do not evolve in a usable way, the potential of virtual and augmented reality might not become fully realized, and the use of VEs will remain accessible only to specialized users. There is a risk that only a small group of authors will possess the necessary skills to create content and design VEs. This would directly impact the costs of creating a VE; the sinking hardware costs would not compensate the increasing authoring costs. But the consequences would be not only purely economic. Many applications need flexible authoring solutions. For instance, in the training application example it is important that trainers are able to modify and adapt the content or UI to their trainees' needs [14]. This calls for the need of dynamically updating the content in a way accessible to the non-specialized user. All VEs are the result of an authoring process, hence VEs in general can benefit from advances in authoring methodologies.

Given its importance, authoring was a topic of the 2013 Dagstuhl seminar Virtual Realities. We report some of the current challenges in the area of authoring identified in the seminar and some of the recent research directions that are being explored in order to address these challenges. We focus on content creation and authoring of UIs for VEs. In the next section, we start by discussing the challenges of recent approaches for authoring tools and processes, and we also discuss the specific challenge of designing immersive interfaces for modeling motions for autonomous virtual characters, which represents a highly relevant example for the next generation type of content creation. In Sections 3 and 4 we discuss novel solutions for approaching these authoring challenges. This is followed by a conclusion.

2 Challenges

The creation of content and UIs that today's VEs require comprises a whole range of authoring activities, from usual multimedia content, to 3D interaction, and even motion specification for simulated autonomous entities. What challenges need to be addressed in order to provide adequate support for these activities? We start by characterizing eight of today's challenges in authoring post-WIMP UIs before we focus on challenges in authoring motions for virtual characters. While promising approaches exist, those challenges are still serious obstacles for using VEs in many real world applications.

2.1 Authoring Post-WIMP User Interfaces

For the creation of post-WIMP UIs, eight challenges were highlighted in the Dagstuhl seminar. We refer to them as the design guideline challenge, standardization challenge, emulation challenge, visibility challenge, authoring process challenge, tool challenge, event aggregation and abstraction challenge, and uncertainty challenge.

The *design guideline challenge* is to collect and to compile experiences and best practice in creating post-WIMP UIs and transform these pieces of information into operational guidelines for interaction techniques to be successfully used in each context and in each use case. User interfaces for virtual environments and generally user interfaces that rely on the post-WIMP paradigm are usually more difficult to design, implement and test than graphical user interfaces (GUIs). One reason is that post-WIMP UIs offer more options how to design a certain interaction. The design space for interaction techniques not only comprises keyboard entries, mouse movements and mouse clicks but for example in-the-air gestures, multi-touch gestures on a surface, speech commands, manipulation of props or movements of the head. Since GUIs have been used much more extensively than post-WIMP UIs, this design space has not been explored as thoroughly; much more experience is available about the design of GUIs and GUI authors have become more proficient by using a limited set of interaction techniques repeatedly. Consequently, there are fewer UI designers accustomed to post-WIMP UIs. Equipping UI designers with the necessary knowledge shows two sides of the design guideline challenge: the deduction of suitable guidelines and processes how to inform authors about these guidelines and support them in their application.

The *standardization challenge* is to agree on suitable common standards for post-WIMP UIs. No established set of interaction techniques comparable to GUI widgets (e.g. radio buttons or sliders) is available. This is not only a problem for UI designers, but also the users often need to familiarize themselves with the interaction techniques they encounter in different VEs. Which standards will emerge? Who will be able to define de facto standards such as the pinch gesture that was made well-known by Apple? While researchers are usually not in the position to set standards, especially de facto standards, they can make valuable contributions in suggesting standards, providing evidence to assess and evaluate standardization proposals, and catalyzing the standardization process.

The *emulation challenge* is to provide the authors with an authoring environment that allow them to test the UI easily on the target platform and switch between the author's view and the user's view. In addition, the authors need to be equipped with methodologies for rapid prototyping to conduct user tests in the early stages of the development (such as paper prototyping that is used for quick GUI prototyping). A reason for this is that the hardware platform the author uses can be significantly different from the target platform. While an author uses a software tool with a GUI in order to create a GUI, an author often does not work in a CAVE or uses body tracking in order to realize a post-WIMP UI. The target platform is used only occasionally for testing, making a WYSIWYG (what you see is what you get) approach impractical. The question is how to emulate aspects of post-WIMP UIs as well as necessary while still provding a comfortable authoring environment and support the author to effortlessly put themselves in the user's position.

The *visibility challenge* is to identify metaphors that are helpful for authoring entities in a user interfaces without visual representation. For authoring GUIs, GUI builders are commonplace and often integrated in development environments such as Eclipse or Visual Studio. GUI builder tools are similar to each other; they offer GUI components to the author together with means for changing parameters of the components (e.g. color, size), combining and arranging them to form a UI. This way, a WYSIWIG approach is realized where the author can inspect the current state of the UI easily and also demonstrate it to users in order to receive a feedback from them. With post-WIMP UIs, however, this approach is not feasible. One reason is that not all components of the UI (e.g. gestures) possess a graphical representation. For example, a thumbs-up gesture could be used to navigate forward in a VE. While there could be a kind of explanatory text or pictorial information present that illustrate to the user how the gesture is to be performed or remind them that this gesture can be used for navigation, this type of graphical represenation is often omitted (e.g. in order to save screen space or to avoid distraction). As a result, a WYSIWIG view gives the author no clue that there is a certain component present in the UI. In GUIs, standard mouse interactions such as the double click have also no visual representation in the UI. But their number is small and they can be enumerated easily. In post-WIMP UIs, a plethora of such interaction techniques might be present.

The *authoring process challenge* is to define appropriate author roles and a workflow outlining the cooperation of authors. Ideally, the creation of a UI is an interdisciplinary effort. During the creation of a GUI, sometimes authors with technical background and programming knowledge and authors with skills in the arts and design work together. In the post-WIMP case, there is even more need to divide the work between several authors with different skill sets [5, 4, 1]. This is due to the increased complexity of the authoring task. For instance, interaction techniques are more difficult to realize [5, 33]. The authoring process needs to provide common ground for the authors to work together seamlessly and to reduce friction. By limiting the prerequisites for an author role, it should also enable new groups of potential authors to contribute to the UI creation (e.g. authors from the application domain who are able to adapt the UI within certain limits).

The *tool challenge* is to support the different authors adequately. Because the tools should allow the author to easily adopt the user's point of view and they should solve the emulation challenge, they need to have post-WIMP UIs themselves. On the one hand, this makes the tools harder to build. On the other hand, post-WIMP interaction techniques might have the potential to make a significant contribution to mastering the tool challenge. Since a new authoring process is needed for post-WIMP UIs, the tools available for GUI creation are not sufficient.

The *event aggregation and abstraction challenge* is to provide events to the application UI programmer on an adequate abstraction and aggregation level. The underlying complexity of the sensor data and its fusion should be hidden from this author role. A significant difference between WIMP and post-WIMP UIs are the number of events that need to be processed in order to react on a user's action. Many sensors (e.g. gyroscopic sensors, depth cameras, tracking devices, pressure sensors) are employed in post-WIMP UIs. They are able to deliver complex information about users (e.g. the pose of different body parts of a user), sometimes more than 1000 times per second. This is in contrast to an occasional 'key pressed' event in GUIs. But the problem is not only the amount of events that need to be processed in real time in order to avoid a lag and to not jeopardize usability. Events need to be aggregated since they may need to be interpreted in context to each other [41]. For instance, the semantics of a hand gesture could differ if there is a certain voice interaction within a time limit.

Moreover, the application UI programmer might find it overwhelming to evaluate hundreds of pieces of information on how users held different parts of their fingers over the time in order to identify whether one of the users performed a 'high-five gesture' or not. Reacting on a single event that has been aggregated from low level events and that states which user performed this gesture is preferable.

The *uncertainty challenge* is to inform the UI programmer about the probability that a specific event has occurred and to equip the author with strategies how to handle the uncertainties. While it can be determined with nearly 100% certainty whether a key was hit on the keyboard or not, this is often not the case with other input devices. Uncertainty might be introduced e.g. due to noise, sampling errors, and sensitivity of sensors to changes in environmental conditions or users making ambiguous gestures.

That these challenges do present obstacles in today's authoring of VEs does not mean that these challenges are exclusive for VEs. For example, the standardization challenge was already present before graphical user interfaces established themselves. But the challenges may raise in VEs and solutions found to those challenges in GUI authoring or authoring of 3D user interfaces may not be simply transferable to post-WIMP UIs. While the uncertainty challenge, for instance, is also an issue in GUI authoring (e.g. do two mouse clicks of the user constitute a double-click or were they meant to be separate clicks), the higher number and complexity of events in post-WIMP user interfaces aggravate this challenge.

2.2 Authoring Motions for Virtual Characters

While the many challenges listed in the previous section address a broad range of needs when designing user interfaces, specific authoring metaphors have still to be developed for types of content that are not easily defined with usual tools. One important example is the definition of motions to be executed by virtual characters.

Improved tools for programming virtual characters are increasingly important in many applications of virtual environments. The challenges described in the previous section still apply to the several sub-problems involved, for ex: definition of agent behaviors, placement and customization of scenarios, etc. However, one particular new challenge that emerges is to design new approaches, techniques and algorithms that can enable new paradigms of user interaction to become effective user interfaces. While such a challenge can be classified as part of the tools challenge, the involved processes and techniques can become significantly complex to address modern still-emerging applications and solutions, such as the problem of programing motions of virtual agents by demonstration. We use this particular problem to illustrate the high complexity that can be involved in a particular challenge of our overall challenge classification.

The process of programming the motion of virtual agents by direct demonstration is highly important because it opens a complex task to the generic user. An effective interface will enable the creation of virtual characters that perform motions as needed in a given application. Due to the great need of variations and precise control of actions and gestures in many scenarios, modeling and parameterization of realistic motions for virtual agents become key problems to be addressed.

Common solutions to motion modeling rely on either hand-crafted motions [63, 20, 60] with commercial modeling tools or algorithmically synthesizing gestures with algorithmic procedures such as Inverse Kinematics [31, 28]. However it remains difficult to achieve both controllable and realistic results, and every attempt to solve the problem purely algorithmically will require specific adaptations and models for every new action and situation being modeled.

On the other hand, motion blending techniques with motion capture data [53, 54, 32, 49] provide powerful interpolation approaches for parameterizing predefined example animations according to high-level characteristics. While intensive research has been dedicated to find suitable interpolation schemes and/or motion style controls, less attention has been given to the development of techniques that enable intuitive interfaces for building suitable motion databases interactively, and that can well cover the simulated workspace with dedicated blending and parameterization procedures. This is especially important for tasks that require parameterizations with respect to spatial constraints within the environment. For instance, the interactive construction of real-time motion controllers has been proposed before [11], but without the inclusion of immersive interfaces for interactive edition and visualization of the obtained models during the creation process.

The obvious approach to this challenge is to develop motion modeling interaction metaphors with mechanisms that are similar to how people would demonstrate motions to each other. Such modeling by demonstration approach is necessary for achieving accessible interfaces that allow generic users to define new motion content. Furthermore, given the recent developments in effective and low-cost 3D sensing (for example with the Kinect) and 3D vision, modeling by demonstration can now be seen as a highly feasible approach to any system.

The recently introduced immersive modeling solution by Camporesi et al [9] implements an interactive motion modeling approach via direct demonstration using an immersive multi-tile stereo visualization systm with full-body motion capture capabilities. The system can be operated in two distinct phases: in the *modeling phase* the expert, who has the specialized knowledge of how to correctly perform the required motions, will demonstrate the needed motions to our system interactively. Later, in the *training phase*, by relying on the database of motions previously collected from the expert, the virtual human trainer is then able to reproduce the motions in interactive sessions with apprentice users learning the training subject, with the ability to reproduce the motions in respect to arbitrary target locations inside the environment.

While this overall approach has the right elements for being very effective, achieving intuitive interactive modeling interfaces has been critical for proving the system to be useful in concrete applications. In particular for the motion modeling phase, intuitive interfaces are important to allow expert trainers to focus on the motions being demonstrated rather than on irrelevant details. A good set of tools for inspection, parameterization and testing are also very important in order for users to effectively build suitable motion databases. Existing motion capture interfaces hardly offer any of these features, and in most cases the user is required to work tedious hours in front of a computer using the keyboard and mouse to perform post-processing operations after many motion capture session.

The needed technology for achieving such types of systems is currently available. As an example Figure 1 exemplifies two prototype configurations being tested at the UC Merced Motion Capture and Visualization facility. It is now the right time to start identifying the principles of a new class of interaction operators and procedures that can achieve intuitive full-body motion demonstration, coverage inspection, parametrization, motion refinement, etc. Section 4 summarizes some of the approaches being investigated in this area.

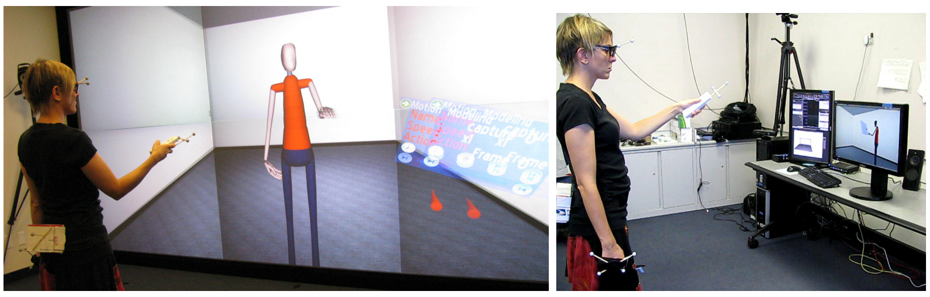

Fig. 1. Example configurations for full-body motion demonstration platforms, from a fully immersive system (left) to a desktop-based environment (right). In both scenarios the demonstrated motions are captured from a reduced marker set placed on the user and tracked by an optical multi-camera tracking system. The motions are then retargetted in real-time to the avatar in order to allow interactive construction of the motion databases to be used by the virtual character [9]. Several variations on these example configurations are possible and customization should be explored according to the goals of a given specific application (see for instance Table 1).

3 Authoring Tools and Processes for Post-WIMP User Interfaces

We present some approaches how to address all the challenges identified in section 2.1. We do not restrict ourselves to 3D and spatial user interfaces [7]. These can be considered a sub-class of the more general post-WIMP UIs. Related to this, we do not require VEs to be fully immersive with using 3D input and output devices. Elements of 2D UIs can become part of a VE [39]. In addition, VEs can be presented as desktop VR, i.e. on a 2D display (e.g. a tabletop display) and still achieve a characteristic goal of a VE: making the user feel present in the virtual environment [46]. As a result, while being able to address authoring issues for VEs, the approaches presented are not exclusively applicable to the authoring of VEs. As post-WIMP UIs are becoming more main stream (see section 1), this could enable research in authoring of VEs to take advantage of the resources and the developments in the area of post-WIMP UIs in general. We start by providing an example of an authoring process, before we discuss frameworks and tools that have been proposed in the literature. Finally, we review some middleware solutions for the challenges.

3.1 An Example of an Authoring Process

One modern example of an authoring process for post-WIMP UIs was presented at the Dagstuhl seminar Virtual Realities: the EMIL authoring process [43] depicted in Fig. 2. It follows the software engineering approach of an object-oriented application framework [17]. The basic idea is to provide an application framework that is initially "'empty"' and needs to be filled with reusable components that are collected in a component library. Components foster reuse. This is not only economical and supports robustness but it is a strategy with regard to the standardization challenge. In addition, there are always two author roles involved when using a component: the author who created it and the author who used it to fill the application framework. Hence, the usage of components fosters the division of work between authors who may possess different skill sets. Consequently, there are two different tools in EMIL: the component authoring tool and the application authoring tool. In EMIL, a component represents an interaction (e.g. a multi-touch gesture, a GUI widget or the interaction facilities of a prop).

Application frameworks in EMIL are tailored to a specific application (e.g. for financial counseling in a bank). They all have a base framework in common that is built on some middleware (e.g. Adobe Flash). It offers an abstract interface for handling user interaction and also for accessing the operating system and the hardware. The base framework implements a mechanism that allows an easy insertion of a component and its linking to the framework. Different author roles are involved in creating the base framework and extending it to application frameworks. All frameworks are always executable. This is in contrast to approaches where components are considered building blocks that need to be assembled first before runnable software is formed. Thus, the current state of the application can always be demonstrated and tested with users even in the very early project stages since every project starts with an empty but working application. Involving users in all stages of UI development and progressing in an iterative manner is characteristic for user-centered design (ISO standard human-centered design for interactive systems ISO 9241-210), today's standard for authoring UIs in general.

EMIL uses a component approach not only for developing software but also for collecting information and fostering organizational learning. Therefore, every interaction component is equipped with a meta-data component. While in the application framework, the meta-data component executes code in order to

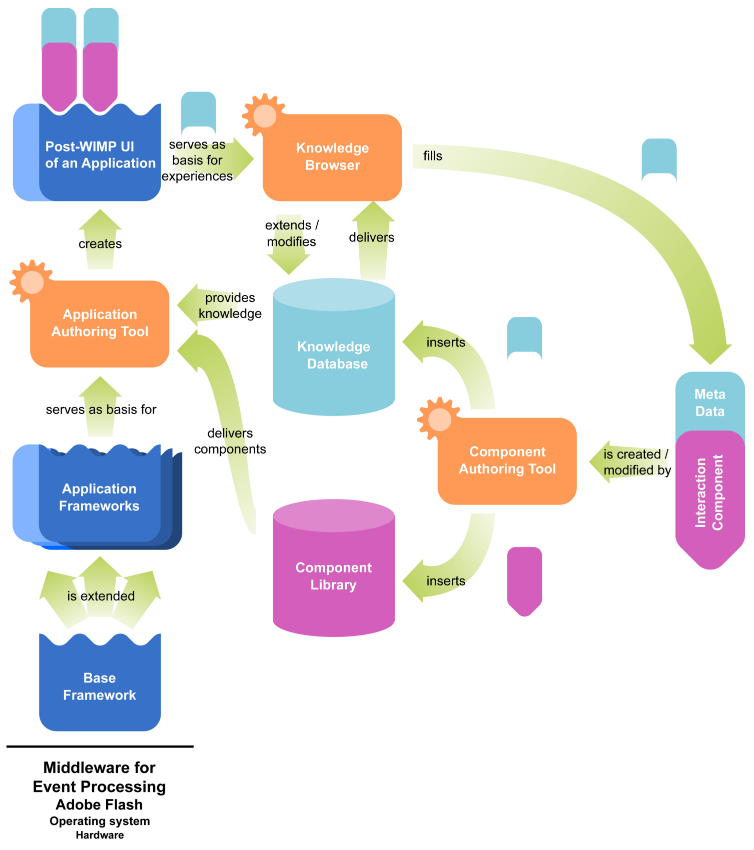

Fig. 2. The EMIL authoring framework for post-WIMP user interfaces

collect information automatically, e.g. it counts how often an interaction component was used, it records user errors that were detected or it measures lag times. The knowledge browser is a tool that allows to import this data and to visualize it to an author. Moreover, with this tool an author is able to prepare an information sheet about this component in the form of an interaction pattern [58]. For example, the author can add videos of user tests were the interaction component was tested or give advice for using this component (e.g. concerning the prerequisites for using the component or other components that have been used successfully in combination with the component). These patterns can be arranged hierarchically in order to form an interaction pattern language analogous to [51]. In the knowledge browser, the author can create a new pattern or augment an existing one that was already stored in the knowledge database. In the application authoring tool, the author who is filling the application framework with interaction components can access the knowledge data base in order to search for and to identify interaction components that are potentially suitable for a given task. On the contrary, this author could also choose an interaction component first. Since this component possesses meta-data, the author would be able to access without searching all the experience already gathered by using this interaction component in other projects. Thus, the access to the knowledge about an interaction component is integrated seamlessly in the authoring process. Using an interaction component multiple times and collecting and refining its meta-data leads to the emergence of design guidelines. This is a strategy to address the design guideline challenge.

Component based approaches only pay off if the components are reused frequently and if there is a critical mass of components available in the library. Considering the diversity of post-WIMP UIs, this might pose a problem. One solution adopted in EMIL is to introduce a specific iteration at the beginning of the iterative development process. This iteration is executed not on the target platform of the post-WIMP UI to be created but on a carefully chosen platform that supports a whole class of post-WIMP UIs, e.g. interactive surfaces or virtual environments. Thus, components and application frameworks created for this class can be reused in different projects in order to illustrate and discuss different UI design alternatives by building prototypes rapidly. During the evaluation of EMIL, this was demonstrated to a team of experts in order to collect qualitative feedback. This team consisted of a UI designer, a programmer and a concept designer from an advertising agency versed in creating post-WIMP UIs for interactive surfaces. They highlighted the ability of this approach for quick design - test cycles with higher quality as even sophisticated interaction techniques can be tried out realistically (e.g. in contrast to a paper prototype). Particularly, they liked that authors involved in the design process can use this prototyping platform to collaboratively assemble and modify the results of their work directly (see Fig. 3 a). Providing each participant with a mobile device is an approach to allow each author to have an individual author's view of the UI (see Fig. 3 b). For this, a magic lens metaphor [6] can be used. This means that the user can apply a visual filter to reveal additional information. It can be implemented by using Augmented Reality methdologies [40].

Moreover, the experts saw an application area in project acquisition where customized prototypes could be used to convince a potential customer of the advantages of post-WIMP UIs [43]. The prototypes can also be used for analyzing the user requirements and for first user testing. The resulting well specified UI can then be realized on the target platform. This is a step towards meeting the emulation challenge.

3.2 Frameworks and Tools

In the literature, several frameworks and tools have been presented specifically for the authoring of virtual reality and augmented reality applications. For example, AMIRE [2] is a component-based framework for authoring augmented reality. It draws from the idea of 3D components [15] that combine graphical representation and behavior. APRIL [34] is a framework for creating augmented reality presentations. Sinem and Feiner describe an authoring framework for

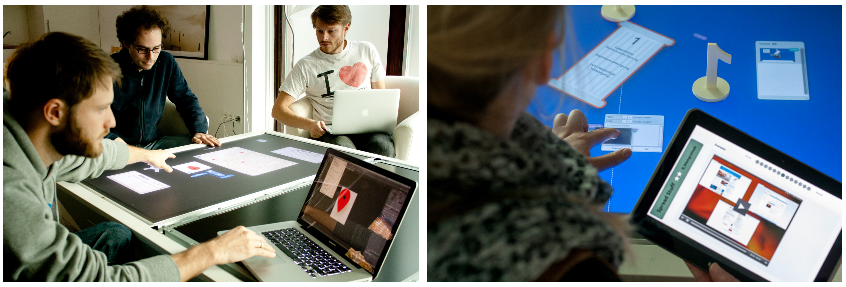

Fig. 3. a) Collaboration of authors when using EMIL rapid prototyping b) Usage of mobile devices for an author-specific view [43]

wearable augmented reality [59]. These frameworks, however, focus on solutions for their specific application field. Moreover, the literature contains reports about methodologies that can be applied to authoring, for example Lee et al. [35] show that it is beneficial to perform the authoring of a VE in an augmented reality environment. Case studies (e.g. [65]) that highlight some of the challenges in authoring are rare.

For authoring post-WIMP UIs several frameworks have been proposed. Some are targeted at certain authoring aspects. For instance, the focus of the OpenInterface (OI) Framework [44] lies on the flexible usage of several input channels for realizing multi-modality. OI is based on components. These can be used in an authoring application where the author manipulates a graphical representation of the user interface based on a data flow graph. Other frameworks focus on specific methodologies. For example, the Designer's Augmented Reality Toolkit (DART) [45] is a framework supporting design activities in early project stages of creating user interfaces employing methodologies from Augmented Reality. In addition, DART aims to assist authors in a smooth transition from low-fidelity prototypes to final products. One of DART's key ideas is to provide behaviors that can be easily attached to content. DART is one of the few approaches where the authoring process is taken explicitly into consideration. More often, not processes but tools are conceived. A tool always shapes an authoring process indirectly.

Many authoring tools used for creating post-WIMP UIs are either used by programmers or by artists. For programmers, text based integrated development environments (IDEs) are commonplace, e.g. Microsoft's Visual Studio [48] or Eclipse [16]. Some tools for non-programmers are available which often employ visual programming techniques [29]. A typical metaphor used is that of a wiring diagram known from electrical engineering. Following the component idea, a set of building blocks is made available. They possess input and output slots and the author's task is to connect them. Examples for such tools are Virtools [64] and Quest3D [50]. Some promising results for such authoring tools are reported, for instance [22] describes an AR authoring tool targeted at non-programmers where tasks can be carried out more accurately and in less than half of the time compared to standard IDEs. The work of Broll et al. shows that visual prototyping can be useful for prototyping VEs [8]. However, these tools are not used widely.

Some IDEs offer functionality for authoring post-WIMP UIs. For example, Apple's Xcode [3] is more than a standard GUI builder. It offers some typical multi-touch gestures that can be connected to the GUI. The connections are visualized resulting in a better overview [38]. Some IDEs can be extended by plugins. For instance, Gesture Studio [41] is a plugin that introduces new authoring metaphors or paradigms to Eclipse, e.g. programming-by-example [55] where the system is trained to recognize gestures by examples provided by the author. However, all these authoring tools are limited to a subset of post-WIMP interaction techniques. A small number of tools are tailored for creating VEs, for example Autodesk's VRED. In addition, tools initially intended for computer game development, e.g. Unity 3D [62] or the sandbox editor of CryEngine3 [12], are also equipped with specific functionality for creating the user interface of VEs.

Only few tools offer comprehensive post-WIMP authoring. Squidy [30] is a tool supporting visual programming under a unified working space. It uses a metaphor based on data flow diagrams as well as a zooming metaphor to provide information about components of the UI on several levels of detail. However, Squidy focuses on technical issues such as the seamless connection of several input and output devices. For example, it is not even possible to create a GUI. A zooming metaphor is also supported by ZOIL (Zoomable Object-Oriented Information Landscape) [67], a software framework for creating distributed post-WIMP UIs. The work in the literature highlights the shortcomings of traditional IDEs and the advantages of providing a dedicated authoring tool for post-WIMP UIs. Still, tools are lacking that are able to address the whole range of post-WIMP interaction techniques. The specific requirements of different authoring roles and the need for supporting collaboration between authors are sometimes not addressed at all.

CLAY (Cross Layer Authoring Tool) [21] explicitly acknowledges that there are different author roles. It distinguishes several layers where each layer represents the work space of one author role. In CLAY, these author roles are programmer, GUI designer, post-WIMP UI designer and end user, i.e. the tool can not only be used during the development of the UI but also for configuration and adaptation after deployment. A slider is used to switch between the layers (see Fig. 4). The slider can also be set between two layers. Transparency is used to blend between these layers making it feasible to view two layers in parallel. Using the same tool and not switching between tools, it is therefore possible to visualize the interfaces and dependencies between the work spaces of different authors, e.g. programmers working on the code layer can see for which GUI widgets (on the GUI layer) and which gestures (on the post-WIMP layer) their methods are used as a callback. This supports collaboration between authors who are assigned to different authoring tasks while working on creating the same post-WIMP UI. Pursuing the idea of a single tool supports a move-

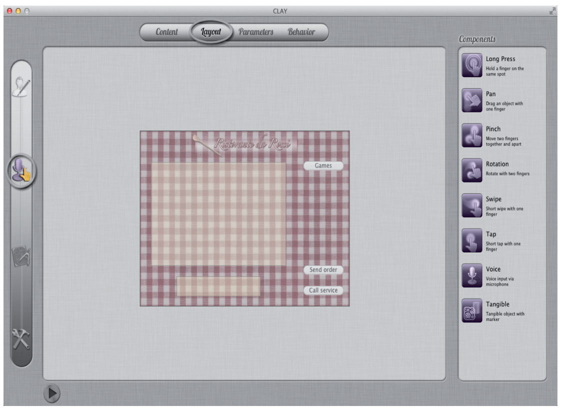
CLAY
Content
Layout
Parameters
Behavior
Components
Long Press
Hold a finger on the same spot
Pan
Drag an object with one finger
Pinch
Move two fingers together and apart
Rotation
Rotate with two fingers
Swipe
Short wipe with one finger
Tap
Short tap with one finger
Voice
Voice input via microphone
Tangible
Tangible object with marker
Ristorante da Rossi
Games
Send order
Call service

Fig. 4. Screenshot of the CLAY authoring tool [21]

ment back and forth as the slider to switch between layers can be moved in two directions. This is more suitable for user-centered design, where results of an evaluation (e.g. a user test on the user layer) need to be fed back to all layers. For example, the need to change a gesture can impact the code. Supporting the different author roles with a single tool can solve problems due to incompatibilities and the need for data exchange between traditional tools. In CLAY, the authors place entities (e.g. widgets or graphical representation of gestures) on the layers and may combine them to form more complex entities. They can also change individual parameters of the entities. CLAY is based on a matrix metaphor, i.e. functionality is arranged in a tabular layout analogous to a matrix. The rows represent different layers, e.g. a code layer or a post-WIMP layer; the columns represent topics, e.g. content, layout, parameters, or behavior. For example, the author can select a cell in the table representing the layout of the GUI layer. In this cell, all according functionality is made available. While this metaphor helps to reduce complexity (since it organizes the functionality and sorts it for different author roles such as a GUI designer), it uses much screen space and can only show connections between two neighboring layers.

Utari3D (Unified Tool for Authoring Realtime Interaction in 3D) is also relying on layers. In contrast to CLAY, it is based on a 3D metaphor and utilizes a specifically shaped 3D space where layers are arranged in parallel. The 3D metaphor provides affordances for possible interactions of the author. In addition, it suggests visualizations. Authors can use a slider to move the layers back and forth. Transparency is used to blend from one layer to the next. In addition, authors can change to a pre-set camera position that allows for a birds-eye view of the layers (see Fig. 5). In the space between the layers, dependencies are visualized. For each switch, a transition animation is used. Utari3D has been used in a user test where several experts built a post-WIMP UI. The evaluation shows that the test users had no difficulty grasping the metaphor. Indeed, these users appreciated an overarching metaphor tying together multiple tools for authoring;

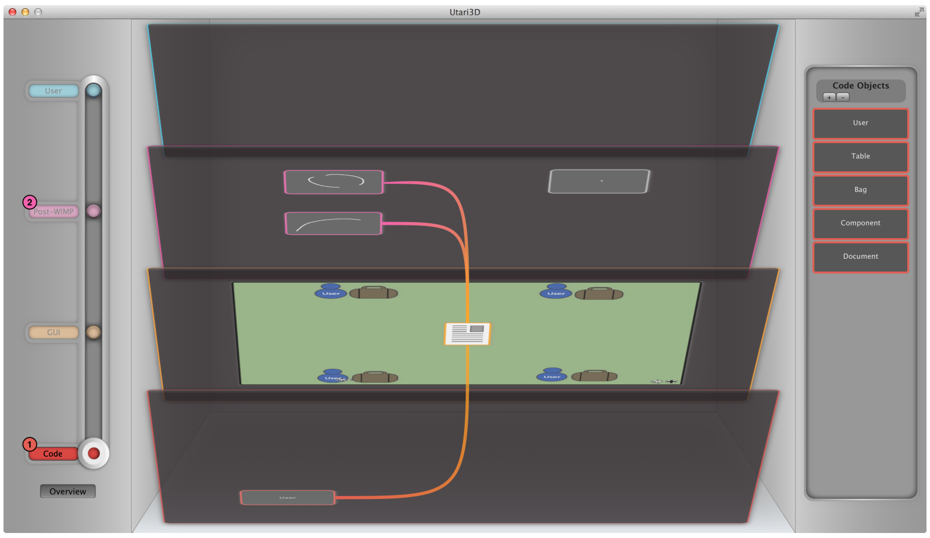

Fig. 5. The overview mode in Utari3D

this is more than just plug-in tools together. The 3D space offers more degrees of freedom to place the camera. Hence, the authors can adopt different views, e.g. look at layers perpendicularly and compare two layers or use the overview mode where they can look in between the layers from a bird's eye perspective. Animating the camera or transforming the layers' position and orientation, smooth transitions can be made between the views. Similar approaches are used for example in Apple's iOS 7 where all browser tabs can be viewed in a 3D space or the cover flow metaphor that allows for browsing music. There is no sudden break by switching between tools which might be confusing and makes it difficult to understand dependencies from entities in one tool to entities in a separate tool. The metaphor can be extended to use 3D layers which is particularly relevant to author the spatial aspects of a VE. Authoring approaches for specific tasks that have proven to be successful and that are used by many authors today (e.g. text-based authoring of program code or visual layout of GUI widgets) can be integrated seamlessly in Utari3D as it is used only for a high level of abstraction. For instance, Utari3D does not impose that every authoring aspect needs to be accomplished graphically and that the use of visual programming techniques is mandatory on the code level. The users in the test appreciated that familiar authoring processes have been complemented and not replaced. Thus, the tools presented do not only address the authoring process challenge, but also the tool challenge.

3.3 Middleware Solutions

In order to address the event aggregation and abstraction challenge and uncertainty challenge often middleware solutions are proposed. Several approaches have been already explored. For instance, the Midas [57] / Mudra [26] system supports the processing of event streams from different sources ranging from low-level, simple raw data streams to higher-level, complex semantic events. It employs a fact base and also a declarative language to specify how facts, i.e. events, are automatically processed. However, this approach has several drawbacks. First, with every event the fact base needs to be constantly updated. As input events are usually short lived and transient in nature this approach requires additional maintenance functions which have to be added to the fact base. Second, the processing rules are realized using the inference engine CLIPS [56], an expert system tool. The inference engine is also not designed for continuous evaluation and thus has to be also modified. Third, the interface to the application layer does not support selection of events based on application defined rules (such as complex Boolean expressions). This is a prerequisite to deal with large amounts of events on different abstraction levels. Lastly, this approach has difficulties in reaching an acceptable performance. Instead of a fact base, [19] used a multi-agent architecture [66] for analyzing all input modalities to find out what the user actually wants to accomplish. While this might be a general approach (such as the Open Agent Architecture [10]), this work focuses on speech input that is accompanied by other types of sensor data in order to resolve ambiguous situations. It remains questionable if this can be transferred to other kinds of post-WIMP interactions. The approaches mentioned above are domain specific approaches to the challenges in the field of HCI. A more general approach is taken by complex event processing (CEP). The term CEP refers to the detection, analysis, and general processing of correlated raw or simple events and their transformation in more abstract and meaningful complex events [42]. CEP is based on the concept of the event driven architecture (EDA), in which loosely coupled components communicate by emitting and consuming events. The methodologies of CEP and EDAs have rarely been applied to interactive systems [23].

UTIL [36] is a middleware that is based on CEP, in particular the interaction event processing architecture [37] and the CEP engine Esper. It applies CEP to interactive systems and provides an abstraction between code of an application's UI and the input devices of a system. It also provides an interaction layer that manages the access of applications to any event from the event processing layer. This middleware can be used for instance as a foundation for the EMIL base framework (see Fig. 2). UTIL offers several benefits. Detection, aggregation, fusion and selection of input events is based on a declarative specification (using the Event Processing Languange EPL). This declarative approach facilitates the specification of post-WIMP interactions on a high level of abstraction. Moreover, it lets the CEP engine perform the actual processing and maintenance of current and past events. This abstraction also makes it possible to benefit from optimizations and other improvements the engine can perform without changing the application's code. Continuous queries enable applications to precisely specify which kinds of events are expected. This allows reducing the amount of post-processing within user interface code to a minimum, thus possibly improving the software quality. It is also feasible to write rules that determine the uncertainty of an aggregated event based on the uncertainty of its base events.

UTIL was evaluated in a representative usage scenario [36]. In this scenario, 4,384 raw input events and 27 rules were used, implementing various gesture-based interaction techniques. In order to also measure the impact of the event delivery, 1,000 unspecific queries for all events on the interaction layer were added. For each raw event passed into the middleware, the time it took the middleware to accept the next event was measured. This round-trip time includes all query evaluations as well as calling all registered event listeners. For the evaluation, a notebook computer (MacBook Pro with an 2.66 GHz Intel Core 2 Duo with 4 GB of RAM) was used. An average roundtrip time of 12.35 μs per event with a standard deviation of 4.8 μs was measured, i.e. an average event throughput of 80,973 event/s. This can be considered sufficient for post-WIMP UIs, as input event rates typically range from hundreds to a few thousand events per second.

4 Motion Modeling Interfaces for Virtual Environments

Among the many functions a user interface may have for content preparation in virtual environments, one that has particular potential to take advantage of a post-WIMP paradigm is the process of programming the motions of virtual agents in virtual environments. With the increased availability of motion capture devices, motion modeling has the clear potential to evolve from an expert-only tedious activity relying on specialized modeling software to a modeling by direct demonstration approach accessible to most users.

Such an approach is extremely important for allowing users to update their content and achieve effective virtual characters in their simulated environments. One particularly important class of applications is the motion modeling of virtual trainers and assistants that can learn, train and assist people in interactive applications of virtual environments. In these applications, facilitating the process of motion content creation is very important. The topic of motion modeling by demonstration represents a good example of a non-traditional type of interface that modern applications are starting to address. This section discusses possible solutions to address this area.

A generic system for interactive motion modeling will have typically two distinct modes: a capture interface and a motion modeling interface. When the user demonstrates new motions on-the-fly during capture, the user can then immediately playback, crop, reject or accept each captured motion segment. This is similar to recording video sequences from a video camera and preparing the sequences for later editing. The motion modeling mode will allow the user to organize the captured motions according to the intended use.

The focus here is on the particular organization of motion segments that is suitable for motion blending and parameterization. The approach creates parameterized gestures or actions from clusters of aligned example motion segments of the same type, but with variations with respect to the spatial parameterization to be considered. For example, a gesture cluster for a certain way of pointing will typically consist of several examples of similar pointing gestures, but with each pointing motion pointing to a different location. Different motion blending techniques can be used with motion capture data in order to achieve parameterization, and a recent comparison study provides a good overview of the most popular methods [18]. The solutions presented on this chapter are based on the *inverse blending* technique [27, 18].

One first issue that has to be addressed by the interactive system is to provide appropriate feedback of the coverage achieved by the current cluster of motions and the employed motion blending technique. The goal is to allow the user to quickly observe the coverage of the database inside the target virtual environment with different visualization tools, and to allow the user to improve the database coverage as needed. The coverage of a database here refers to how well parameterized motions interpolated from the discrete motion examples in the current database are able to satisfy precise spatial constraints as needed in the environment. With the appropriate visualization tools the user is able to quickly switch between capture and modeling modes until the needed coverage is achieved, therefore guaranteeing correct execution in the target environment.

Figure 6 shows one example scenario where the user is modeling a pouring motion cluster with a *motion vest* motion capture interface based on inertial sensors. For the illustrated pouring action the spatial parameterization needs to well cover the target container, which can be placed anywhere on the table in the scenario. By providing a few pouring actions for key locations on the table inverse blending procedures can then precisely interpolate the given example motions towards arbitrary targets on the table.

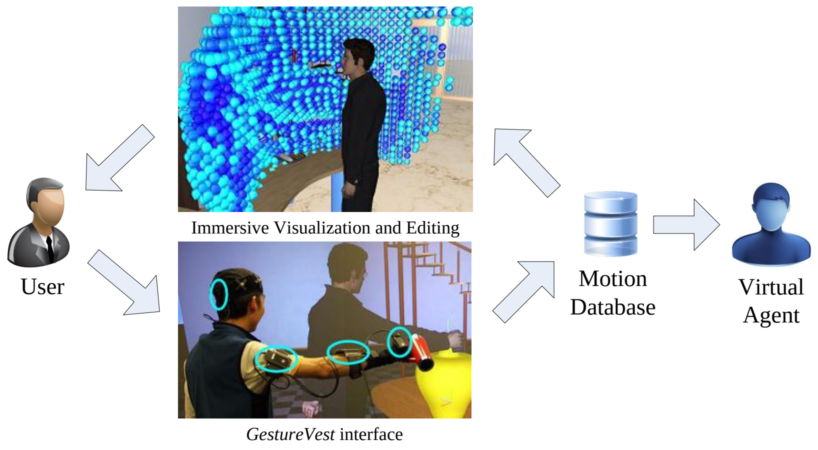

Fig. 6. The main components for an interactive motion modeling by demonstration system.

The definition of clusters for collecting example motions is an important concept of system based on motion blending. Clusters are necessary for specifying each parameterized action or gesture. When the user selects to start a new cluster, every recorded motion becomes associated with that cluster. Motions in a cluster will be blended and therefore they have to consistently represent variations of a same type of motion. The capture process is straightforward and it just requires a button to notify start and stop signals. The user first initializes a new motion cluster, and then holds down the capture button to begin recording. A button in the WiiMote controller is used in the presented examples. The button is then released after performing the motion.

Using the WiiMote controller, the user can instantly switch from on-line capture mode to playback/editing, scroll through the already captured motions, delete unwanted clips from memory, mark the start and end of each motion segment to crop out unnecessary parts, and most importantly mark the stroke frames (stroke times) for each segment before populating them into clusters in the database. The stroke frame of each captured motion is then used as the point in the motion (the main stroke point) to be parameterized.

4.1 Motion Database Inspection

Since great variations can be found among different gestures and actions, an easy way to go through each motion cluster and to quickly crop lead-in/lead-out frames and annotate the motion strokes is needed. We have experimented with buttons to skip over motions captured and with an editing interaction mode where the trajectory of the user's hand is captured and mapped into a linear horizontal movement in real-time, and the movement directly controls the motion playback slider. This enables the user to quickly visualize the available motions with a simple horizontal hand movement, allowing the user to remain inside the capture area and conveniently validate and edit the motion cluster. Figure 7 shows several snapshots of the playback interface being used to analyze a recently captured pointing motion.

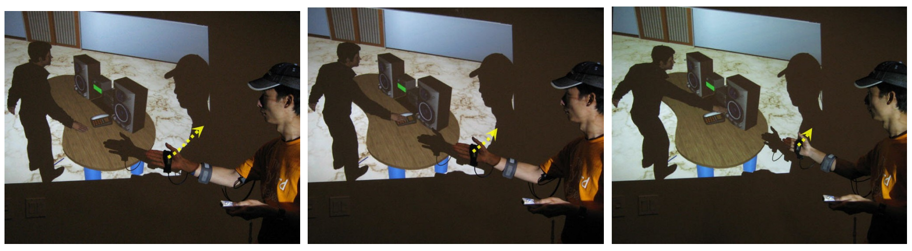

Fig. 7. This example interface allows the user to easily scroll through captured motions, crop out unwanted frames, and most importantly mark stroke frames to be parameterized. This image shows the user inspecting one motion segment with a simple hand movement.

4.2 Workspace Coverage Visualization

The ability to enforce constraints using inverse blending greatly depends on the existing variations among the example motions being interpolated. In general, the size of motion database is proportional to the volume of the covered workspace. In order to produce quality motions satisfying many possible constraints spanning the whole workspace, it is important to determine which example motions to capture during the capture process. This will ensure that a well-built cluster of motions is formed, with good coverage of the regions of interest (ROIs) inside the workspace.

On the other hand, defining an overly fine subdivision of the constraint space with too many examples is inefficient and impractical as it requires capturing too many example motions to populate a database. Not only the database would be redundant, this would also impose a huge workload on the user. Instead, since similar examples can often be interpolated to produce valid new motions with good quality, a small number of carefully selected example motions is better in providing good coverage for the ROIs in the workspace. Achieving an efficient database is also key to ensure lag-free interactivity of the system.

The approach of using a palette of colors to quantify error inside the workspace is explored here in order to intuitively guide the user during the process of adding new motions to the database. A global coverage visualization of the workspace volume can be achieved with a coarse uniform sampling of workspace points. Each point is assigned a color that quantifies the error achieved by the inverse blending routine when addressing the target point.

Consider a pointing database as example. In this case the constraint is the target locations for the finger tip to reach. The user can visualize how accurate the current cluster of motions can be parameterized to cover the entire area in front of the virtual character. This example is shown in Figure 8. In the example, the error measures the distance between each sampled pointing target (small color cubes) and the position that can actually be reached by the finger tip, using the current database. While the visualization may be of a large portion of the workspace, the computed colors come from the interpolation errors in the current database independent of the quantity of example motions in the database.

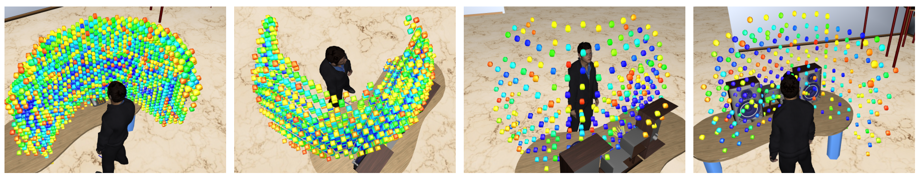

Fig. 8. Volume visualization with global workspace sampling at different sampling densities. The error threshold can be adjusted to only display regions with large errors and thus help the user to focus on improving those regions.

This visualization method requires some computation time, normally a few seconds, depending on the fineness/coarseness of the sampling. After this initial computation, it can be visualized from different points of views interactively in real-time without any lag. This solution provides a good overview of the whole cluster coverage in the given environment.

4.3 Local Coverage Visualization

It is possible to observe that the global error-based volume visualization is not needed when the user is fine tuning the coverage of a small region, or when only a small local region is of interest. In addition, even if using the global cluster coverage visualization is always helpful, the pre-computation time can impose undesirable wait times when editing large motion sets since every time new motions are added to the cluster visualizer nodes have to be re-computed.

A possible second coverage inspection method is local and allows the user to interactively place a colored mesh geometry in the scene for precise coverage visualization in a specific region. See Figure 9 for an example using the same pointing motion database as in Figure 8. When switched to the local coverage visualization mode, the system renders a transparent colored mesh geometry in the approximate form of a sphere covering a small ROI inside the workspace, which follows the movement of the user's hand. In this case, the motion cluster coverage evaluation is performed only within the specific region delimited by the mesh geometry volume.

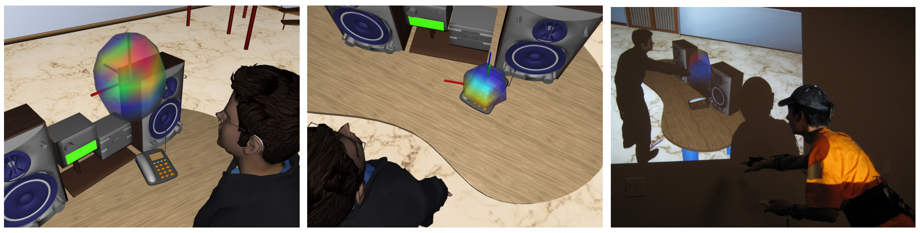
Fig. 9. The local coverage visualizer can be interactively placed over a small ROI during the motion modeling phase, enabling the user to instantly visualize the current motion cluster coverage at important locations. The user can then easily further improve the cluster as needed. The right-most image shows the user interactively placing the local coverage visualizer close to the control panel of the sound system to inspect the cluster coverage in those specific regions.

The mesh color computation is very efficient. Only the vertices of the mesh are evaluated for error computation, and the obtained color on the mesh surface comes from color interpolation with Gouraud shading. Mesh size, shape and resolution can be easily changed during the interactive inspection by using the WiiMote controller. The user can select different mesh sizes for either fast sweeping of areas of interests, or for carefully checking small spots of interests. The local coverage visualization mode is particularly useful for close examination of coverage within small local ROIs. This mode easily takes into account any new demonstrated motions dynamically added to the database without any pre-computation lag.

5 Conclusion

The eight challenges for authoring VEs (design guideline challenge, standardization challenge, emulation challenge, visibility challenge, authoring process challenge, tool challenge, event aggregation and abstraction challenge and uncertainty challenge) identified are still serious obstacles for a more widespread use of virtual environments in various application areas. In addition to these broad challenges, there is a lack of specific authoring metaphors for addressing new emerging types of content and content creation. One highly relevant example is immersive motion modeling, i.e. the authoring of motions to be executed by virtual characters. The techniques discussed in this paper provide an overview of solutions implementing an immersive motion modeling interface.

It is important to observe that for the overall approach of immersive motion modeling to be most effective the involved design choices have to be customized according to the target application. Table 1 provides an analysis of several types of relevant applications and their respective relevant characteristics. Table 1 was built from the experience of the authors working with the described system.

Types of Applications	**Design Choices**						
	stereo vizualization	*scene in 100% scale*	*tracking scope*	*tracking precision*	*motion modeling*	*coverage visualization*	*challenges*
high-precision procedures with object manipulation, ex: virtual surgery training	important	important	full upper-body with fingers	high precision important	direct interaction important	important	VR system alone will not provide force feedback
procedures where object manipulation is not crucial, ex: demonstration of machine operations	maybe important	maybe important	full upper-body with fingers	medium precision may be enough	both direct interaction and avatar display useful	important	need effective interface for virtual objects manipulation
generic demonstrative gestures for delivery of information, ex: giving directions or object info.	maybe important	maybe important	single arm with fingers	medium precision may be enough	both direct interaction and avatar display useful	important	current solutions should be sufficient
modeling generic motions/exercises without reference to objects, ex: exercises for physical therapy	less important	less important	full body no fingers	medium precision may be enough	modeling motions with only avatar display may be more effective	may not be needed	current solutions should be sufficient

Table 1. Analysis of possible types of applications, their main characteristics, and the design choices relevant to immersive modeling interfaces.

Even if the described underlying algorithmic solutions based on motion blending are replaced by other methods, a user-driven motion capture based interface will need to integrate basic motion segmentation editing tools, quick motion database traversal tools, and coverage/quality of results visualization. The discussed examples only provide a first exploration of techniques into this broad realm.

Likewise, the examples for authoring processes, authoring framework and authoring tools for creating virtual environments in general and their user interfaces in particular that were presented can also be considered only first steps towards solving the wide-ranging challenges. More research efforts are necessary in this area and the novel approaches to authoring and modeling need to be employed and evaluated more extensively in field tests. One important issue for the future research roadmap is to not only examine every authoring task and authoring challenge idependently but to provide authors with a consistent and coherent authoring framework.

Acknowledgements Marcelo Kallmann and David Huang were partially supported in this work by NSF award IIS-0915665. Ralf Dörner was partially supported in this work by the German Federal Ministry of Education and Research (BMBF) grant 17043X10.

References

1. Abawi, D.F., Dörner, R., Grimm, P.: A component-based authoring environment for creating multimedia-rich mixed reality. In: Proceedings of the Seventh Eurographics Conference on Multimedia. pp. 31–40. EGMM'04, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2004), `http://dx.doi.org/10.2312/EGMM/MM04/031-040`
2. Abawi, D.F., Dörner, R., Grimm, P.: A component-based authoring environment for creating multimedia-rich mixed reality. In: Proceedings of the Seventh Eurographics Conference on Multimedia. pp. 31–40. EGMM'04, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2004), `http://dx.doi.org/10.2312/EGMM/MM04/031-040`
3. Apple: Xcode (Jan 2014), `developer.apple.com/xcode`
4. Bastide, R., Navarre, D., Palanque, P.: A model-based tool for interactive prototyping of highly interactive applications. In: CHI '02 Extended Abstracts on Human Factors in Computing Systems. pp. 516–517. CHI EA '02, ACM, New York, NY, USA (2002), `http://doi.acm.org/10.1145/506443.506457`
5. Beaudouin-Lafon, M.: Instrumental interaction: An interaction model for designing post-wimp user interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 446–453. CHI '00, ACM, New York, NY, USA (2000), `http://doi.acm.org/10.1145/332040.332473`
6. Bier, E.A., Stone, M.C., Pier, K., Buxton, W., DeRose, T.D.: Toolglass and magic lenses: The see-through interface. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques. pp. 73–80. SIGGRAPH '93, ACM, New York, NY, USA (1993), `http://doi.acm.org/10.1145/166117.166126`
7. Bowman, D.A., Kruijff, E., LaViola, J.J., Poupyrev, I.: 3D User Interfaces: Theory and Practice. Addison-Wesley, Boston, MA, USA (2004)
8. Broll, W., Herling, J., Blum, L.: Interactive bits: Prototyping of mixed reality applications and interaction techniques through visual programming. In: 3D User Interfaces, 2008. 3DUI 2008. IEEE Symposium on. pp. 109–115. IEEE (2008)

9. Camporesi, C., Huang, Y., Kallmann, M.: Interactive motion modeling and parameterization by direct demonstration. In: Proceedings of the 10th International Conference on Intelligent Virtual Agents (IVA) (2010)
10. Cheyer, A., Martin, D.: The open agent architecture. Autonomous Agents and Multi-Agent Systems 4(1-2), 143–148 (Mar 2001), http://dx.doi.org/10.1023/A:1010091302035
11. Cooper, S., Hertzmann, A., Popović, Z.: Active learning for real-time motion controllers. ACM Transactions on Graphics (SIGGRAPH 2007) 26(3) (Aug 2007)
12. Crytech: Cryengine 3 (Jan 2014), cryengine.com
13. van Dam, A.: Post-wimp user interfaces. Commun. ACM 40(2), 63–67 (Feb 1997), http://doi.acm.org/10.1145/253671.253708
14. Dörner, R., Grimm, P.: Etoile - an environment for team, organizational and individual learning in emergencies. In: Proceedings of the 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. pp. 27–34. WETICE '00, IEEE Computer Society, Washington, DC, USA (2000), http://dl.acm.org/citation.cfm?id=647068.715502
15. Dörner, R., Grimm, P.: Three-dimensional beanscreating web content using 3d components in a 3d authoring environment. In: Proceedings of the fifth symposium on Virtual reality modeling language (Web3D-VRML). pp. 69–74. ACM (2000)
16. Eclipse: Eclipse foundation homepage (Jan 2014), eclipse.org
17. Fayad, M., Schmidt, D.C.: Object-oriented application frameworks. Commun. ACM 40(10), 32–38 (Oct 1997), http://doi.acm.org/10.1145/262793.262798
18. Feng, A., Huang, Y., Kallmann, M., Shapiro, A.: An analysis of motion blending techniques. In: Proceedings of the Fifth International Conference on Motion in Games (MIG). Rennes, France (2012)
19. Flippo, F., Krebs, A., Marsic, I.: A framework for rapid development of multimodal interfaces. In: Proceedings of the 5th International Conference on Multimodal Interfaces. pp. 109–116. ICMI '03, ACM, New York, NY, USA (2003), http://doi.acm.org/10.1145/958432.958455
20. Gebhard, P., Kipp, M., Klesen, M., Rist, T.: What are they going to talk about? towards life-like characters that reflect on interactions with users. In: Proc. of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE'03) (2003)
21. Gerken, K., Frechenhäuser, S., Dörner, R., Luderschmidt, J.: Authoring support for post-wimp applications. In: Kotz, P., Marsden, G., Lindgaard, G., Wesson, J., Winckler, M. (eds.) Human-Computer Interaction INTERACT 2013, Lecture Notes in Computer Science, vol. 8119, pp. 744–761. Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-40477-1_51
22. Gimeno, J., Morillo, P., Orduna, J., Fernandez, M.: A new ar authoring tool using depth maps for industrial procedures. Computers in Industry 64(9), 1263 – 1271 (2013), http://www.sciencedirect.com/science/article/pii/S0166361513001267, special Issue: 3D Imaging in Industry
23. Hinze, A., Sachs, K., Buchmann, A.: Event-based applications and enabling technologies. In: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems. pp. 1:1–1:15. DEBS '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1619258.1619260
24. Homepage: Leap motion (Jan 2014), leapmotion.com
25. Homepage: Oculus rift (Jan 2014), oculusvr.com
26. Hoste, L., Dumas, B., Signer, B.: Mudra: A unified multimodal interaction framework. In: Proceedings of the 13th International Conference on Multimodal Inter-

faces. pp. 97–104. ICMI '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/2070481.2070500
27. Huang, Y., Kallmann, M.: Motion parameterization with inverse blending. In: Proceedings of the 3rd International Conference on Motion in Games (MIG) (2010)
28. Kallmann, M.: Analytical inverse kinematics with body posture control. Computer Animation and Virtual Worlds 19(2), 79–91 (2008)
29. Kelleher, C., Pausch, R.: Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Comput. Surv. 37(2), 83–137 (Jun 2005), http://doi.acm.org/10.1145/1089733.1089734
30. König, W.A., Rädle, R., Reiterer, H.: Squidy: A zoomable design environment for natural user interfaces. In: CHI '09 Extended Abstracts on Human Factors in Computing Systems. pp. 4561–4566. CHI EA '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1520340.1520700
31. Kopp, S., Wachsmuth, I.: Synthesizing multimodal utterances for conversational agents: Research articles. Computer Animation and Virtual Worlds 15(1), 39–52 (2004)
32. Kovar, L., Gleicher, M.: Automated extraction and parameterization of motions in large data sets. ACM Transaction on Graphics (Proceedings of SIGGRAPH) 23(3), 559–568 (2004)
33. Lawson, J.Y.L., Coterot, M., Carincotte, C., Macq, B.: Component-based high fidelity interactive prototyping of post-wimp interactions. In: International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction. pp. 47:1–47:4. ICMI-MLMI '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1891903.1891961
34. Ledermann, F., Schmalstieg, D.: April: a high-level framework for creating augmented reality presentations. In: Virtual Reality, 2005. Proceedings. VR 2005. IEEE. pp. 187–194. IEEE (2005)
35. Lee, J.Y., Seo, D.W., Rhee, G.W.: Tangible authoring of 3d virtual scenes in dynamic augmented reality environment. Computers in Industry 62(1), 107–119 (2011)
36. Lehmann, S., Doerner, R., Schwanecke, U., Haubner, N., Luderschmidt, J.: Util: Complex, post-wimp human computer interaction with complex event processing methods. In: Proceedings of the 10th Workshop Virtual and Augmented Reality of the GI Group VR/AR. pp. 109–120. Shaker Verlag, Aachen (2013)
37. Lehmann, S., Doerner, R., Schwanecke, U., Luderschmidt, J., Haubner, N.: An architecture for interaction event processing in tabletop systems. In: Proceedings of the first workshop Self Integrating Systems for Better Living Environments, Sensyble 2010. pp. 15–19. Shaker Verlag, Aachen (2010)
38. Li, P., Wohlstadter, E.: View-based maintenance of graphical user interfaces. In: Proceedings of the 7th International Conference on Aspect-oriented Software Development. pp. 156–167. AOSD '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1353482.1353501
39. Lindeman, R.W., Sibert, J.L., Hahn, J.K.: Towards usable vr: An empirical study of user interfaces for immersive virtual environments. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 64–71. CHI '99, ACM, New York, NY, USA (1999), http://doi.acm.org/10.1145/302979.302995
40. Looser, J., Grasset, R., Billinghurst, M.: A 3d flexible and tangible magic lens in augmented reality. In: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. pp. 1–4. IEEE Computer Society (2007)

41. Lü, H., Li, Y.: Gesture studio: Authoring multi-touch interactions through demonstration and declaration. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 257–266. CHI '13, ACM, New York, NY, USA (2013), `http://doi.acm.org/10.1145/2470654.2470690`
42. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2001)
43. Luderschmidt, J., Haubner, N., Lehmann, S., Dörner, R.: Emil: A rapid prototyping authoring environment for the design of interactive surface applications. In: Proceedings of the 15th International Conference on Human-Computer Interaction: Human-centred Design Approaches, Methods, Tools, and Environments - Volume Part I. pp. 381–390. HCI'13, Springer-Verlag, Berlin, Heidelberg (2013), `http://dx.doi.org/10.1007/978-3-642-39232-0_42`
44. Luderschmidt, J., Haubner, N., Lehmann, S., Dörner, R.: Emil: A rapid prototyping authoring environment for the design of interactive surface applications. In: Proceedings of the 15th International Conference on Human-Computer Interaction: Human-centred Design Approaches, Methods, Tools, and Environments - Volume Part I. pp. 381–390. HCI'13, Springer-Verlag, Berlin, Heidelberg (2013), `http://dx.doi.org/10.1007/978-3-642-39232-0_42`
45. MacIntyre, B., Gandy, M., Dow, S., Bolter, J.D.: Dart: A toolkit for rapid design exploration of augmented reality experiences. In: Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology. pp. 197–206. UIST '04, ACM, New York, NY, USA (2004), `http://doi.acm.org/10.1145/1029632.1029669`
46. McMahan, A.: Immersion, engagement and presence. The video game theory reader pp. 67–86 (2003)
47. Microsoft: Kinect homepage (Jan 2014), `xbox.com/kinect`
48. Microsoft: Microsoft visual studio (Jan 2014), `visualstudio.com`
49. Mukai, T., Kuriyama, S.: Geostatistical motion interpolation. In: ACM SIGGRAPH. pp. 1062–1070. ACM, New York, NY, USA (2005)
50. Quest3d: Act-3d b.v.: Quest3d (Jan 2014), `quest3d.com/`
51. Remy, C., Weiss, M., Ziefle, M., Borchers, J.: A pattern language for interactive tabletops in collaborative workspaces. In: Proceedings of the 15th European Conference on Pattern Languages of Programs. pp. 9:1–9:48. EuroPLoP '10, ACM, New York, NY, USA (2010), `http://doi.acm.org/10.1145/2328909.2328921`
52. Richards, D., Porte, J.: Developing an agent-based training simulation using game and virtual reality software: Experience report. In: Proceedings of the Sixth Australasian Conference on Interactive Entertainment. pp. 9:1–9:9. IE '09, ACM, New York, NY, USA (2009), `http://doi.acm.org/10.1145/1746050.1746059`
53. Rose, C., Bodenheimer, B., Cohen, M.F.: Verbs and adverbs: Multidimensional motion interpolation. IEEE Computer Graphics and Applications 18, 32–40 (1998)
54. RoseIII, C.F., Sloan, P.P.J., Cohen, M.F.: Artist-directed inverse-kinematics using radial basis function interpolation. Computer Graphics Forum (Proceedings of Eurographics) 20(3), 239–250 (September 2001)
55. Rubine, D.: Specifying gestures by example. In: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques. pp. 329–337. SIGGRAPH '91, ACM, New York, NY, USA (1991), `http://doi.acm.org/10.1145/122718.122753`
56. Savely, R., Culbert, C., Riley, G., Dantes, B., Ly, B., Ortiz, C., Giarratano, J., Lopez, F.: Clips (Jan 2014), `clipsrules.sourceforge.net`

57. Scholliers, C., Hoste, L., Signer, B., De Meuter, W.: Midas: A declarative multi-touch interaction framework. In: Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction. pp. 49–56. TEI '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/1935701.1935712
58. Seffah, A., Taleb, M.: Tracing the evolution of hci patterns as an interaction design tool. Innov. Syst. Softw. Eng. 8(2), 93–109 (Jun 2012), http://dx.doi.org/10.1007/s11334-011-0178-8
59. Sinem, G., Feiner, S.: Authoring 3d hypermedia for wearable augmented and virtual reality. In: 2012 16th International Symposium on Wearable Computers. pp. 118–118. IEEE Computer Society (2003)
60. Stone, M., DeCarlo, D., Oh, I., Rodriguez, C., Stere, A., Lees, A., Bregler, C.: Speaking with hands: creating animated conversational characters from recordings of human performance. ACM Transactions on Graphics 23(3), 506–513 (2004)
61. Sutherland, I.E.: The ultimate display. IFIP'65 International Federation for Information Processing 2, 506–508 (1965)
62. Technologies, U.: Unity3d (Jan 2014), unity3d.com
63. Thiebaux, M., Marshall, A., Marsella, S., Kallmann, M.: Smartbody: Behavior realization for embodied conversational agents. In: Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2008)
64. Virtools: Daussault systemes: 3dvia virtools (Jan 2014), 3ds.com/de/products-services/3dvia/3dvia-virtools/
65. Wojciechowski, R., Walczak, K., White, M., Cellary, W.: Building virtual and augmented reality museum exhibitions. In: Proceedings of the ninth international conference on 3D Web technology. pp. 135–144. ACM (2004)
66. Woolridge, M.: An Introduction ot MultiAgent Systems. John Wiley & sons, Hoboken, NJ, USA (2002)
67. Zoellner, M., Jetter, H.C., Reiterer, H.: Zoil: A design paradigm and software framework for post-wimp distributed user interfaces. In: Gallud, J.A., Tesoriero, R., Penichet, V.M. (eds.) Distributed User Interfaces, pp. 87–94. Human-Computer Interaction Series, Springer London (2011), http://dx.doi.org/10.1007/978-1-4471-2271-5_10