ACTIF: An Interactor Centric Interaction Framework

Nicolai Hess* Jan D.S. Wischweh† Kirsten Albrecht‡
Kristopher J. Blom§ Steffi Beckhaus¶
im/ve, University of Hamburg, Germany

Abstract

The design and implementation of interactions in 3D environments remains a challenge. This is especially true for novices. Mechanisms to support the creation of interaction have been developed, but they lack a central metaphor that fits the natural way in which developers conceptualize interaction techniques. In this paper, we introduce a new framework whose design mirrors the essence of interaction throughout the Virtual Reality spectrum, where the user is literally in the center. It also reflects the way in which interactions are actually understood and described, based on the interactor and her actions.

Based on the central metaphor of the interactor, an implementation that is composed of three phases is developed. Those phases are: input retrieval and shaping, interpretation of user intentions, and execution of changes to the environment. Through these divisions, software requirements like composition and reusability of components are satisfied. The resultant system ACTIF, an ACTor centric Interaction Framework, structures interaction development in a meaningful and understandable way and at the same time eases the design and creation of new and experimental interactions.

CR Categories: H.5.2 [Information Systems]: Information Interfaces And Presentation—User Interfaces I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism I.3.6 [Computing Methodologies]: Computer Graphics—Methodology and Techniques

Keywords: 3D User Interfaces, User Centered HCI, Virtual Reality

1 Introduction

Interaction in three dimensional (3D) user interfaces remains a challenging aspect of Virtual Reality and related fields. Mechanisms supporting the creation of interaction with Virtual Environments (VEs) are required to ease both development and exploration of this space, and also to support an expanding growing user base that includes novices. Various approaches have been taken in the development of software to address the support of interaction design and implementation. These methods have largely focused adapting well understood software approaches to 3D interaction or on software engineering aspects, like component reusability. We propose approaching the creation of a 3D interaction framework from a different direction, centered on the interactor.

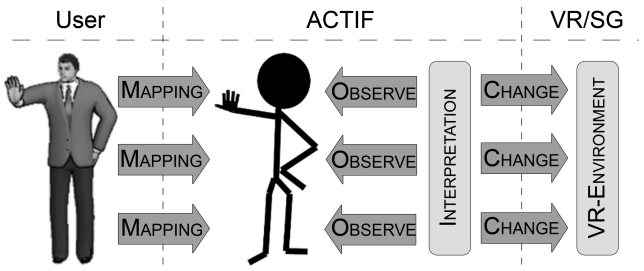

Figure 1: *The basic concept of our design divides the interaction process into three phases built around a Virtual Actor.*

In this paper, ACTIF - an ACTor centric Interaction Framework - is presented. ACTIF is built around the central metaphor of an Virtual Actor, stemming from the observation that the user is literally in the center interface. Therefore, a truly "user centered approach" should be taken. ACTIF is also based on the idea that interactions are initially understood with respect to what the interactor does. In traditional systems, that understanding must then be translated into the representation of the given framework.

The Virtual Actor is the central unifying component of the ACTIF interaction system. Building a system around this central point, three different aspects of an interaction implementation were identified. These phases of the interaction are: input retrieval and shaping, interpretation of user intentions, and execution of changes to the environment. The structure of this approach to interaction specification is outlined in Figure 1. Applying this approach reveals a different way to achieve the traditional requirements of interaction frameworks, like consisting of componentized units that can be reused and separation of system specifics.

This paper presents the development of the ACTIF interaction framework, a new approach to supporting interaction development. The next section reviews the existing approaches to this problem space. The design and implementation of ACTIF are presented in Section 3. The ACTIF system is discussed in Section 4, before concluding the paper.

2 Background

Various methods for supporting interaction via specialized software have been explored. Three basic methods can be identified from those works: scene graph coupled manipulators and independent frameworks that are either dataflow based or event based.

Many of the monolithic VR systems include interaction support integrated into their scene graph representation. The most prevalent is the approach taken by the SGI Inventor system [Strauss 1993]. Special manipulators that work in conjunction with extended scene graph nodes to define interactions. These systems start every frame with special "Sensors" that insert input from the world. A manipulator receives the input data and is responsible for converting the input into a change in the environment.

*e-mail:nicolaihess@web.de
†e-mail:mail@wischweh.de
‡e-mail:Kirsten-Albrecht@gmx.de
§e-mail:blom@informatik.uni-hamburg.de
¶e-mail:steffi.beckhaus@uni-hamburg.de

The dataflow based approach models interaction as a series of operations on the data as it flows from the device interfaces to components that actually change the graphics. Each operation is a sort of filter that changes the data. The advantage of the dataflow is that the filters can be created such that they build minimal components and can be reused in different contexts. A current example of such a system is the InTmL framework [Figueroa et al. 2001]. An alternative approach to the dataflow system is an event based system. Such event based systems are similar to the standard method used in desktop based systems. The basic idea here is that some component reacts to events generated by the user input. Kessler's SVIT system is an example of this method [Kessler 1999].

At an abstract level, the work of Wingrave and Bowman [Wingrave and Bowman 2008] is the most relevant work to our approach. They term their approach as being "developer-centric." Although the concrete approach of the Chasm tool is quite different than ours, the approach likewise strives to assist the developer by adding a conceptual structure to the development of the interaction methods. As with their approach, our design strives to assist the developer by structuring the interaction in a way that simplifies the conceptualization and ultimately the development of interaction methods.

3 ACTIF Interaction Framework

The development presented here is the result of and also influenced by the context in which it was created. As part of a course focusing on interaction in VR, a set of students were asked to develop a new framework for interaction under a diverse set of conditions. The overarching goal of the development was to design and build a new method for supporting the creation of interactions. Creating interactions was also part of the course work, which meant that the framework was to be used by the developers themselves in a second stage, as well as by their classmates and even future classes. However, the course also took place in a time of transition, where a change in VR systems was expected, meaning their system had to be very flexible.

The combination of abstraction from input, component reusability, incremental prototyping, VR system abstraction, and teaching other students how the new system worked led to the conclusion that the existing design patterns were not optimal. The introduction of a Virtual Actor to the system unifies and clarifies those concepts, creating a system that fulfills those diverse requirements. With the Virtual Actor as a centerpiece of the idea and the system proposal, a new view of the interaction composition was made possible. The movement from a conceptualized interaction method to an implementation can be approached from the point of the interactor. In this approach, the interactor in the virtual world is found in the center of the interaction implementation.

The ACTIF interaction framework is the result of applying this approach. The central Virtual Actor concept gives rise to a convenient and advantageous method of dividing the interaction into components. ACTIF distinguishes between three different aspects of developing interaction techniques: (a) mapping data from input devices to meaningful data in the virtual world, (b) interpreting what the intention of the user is, (c) taking actions to carry out the users will. This process is depicted graphically in Figure 1.

ACTIF is a C++ library that is cross platform and cross VR system. The implementation is component based, where each step of the process is encapsulated in its own component. The entire system is controlled by a central kernel that enforces this three stage process. The kernel runs registered components each frame. Modal and smart update mechanisms allow ACTIF to implement complex interactions.

In this section we introduce the ACTIF design and implementation in more detail. As the central concept of the system, the Virtual Actor is introduced first. The details of the implementation of the three phases are introduced after that. The control loop of ACTIF is then presented. Finally, the integration of ACTIF into existing VR systems is presented the final portion of this section.

3.1 Virtual Actor

The Virtual Actor plays a central and major role in ACTIF. However, it is rather a simple component. The Virtual Actor is data structure instead of a being an active component. The Virtual Actor is an abstraction of all aspects of the user's behavior. The Virtual Actor is not an avatar, though it may steer one. It is the interactor in the virtual world that is steered by the user in the physical world. Developing interactions in ACTIF is, therefore, about manipulating and observing the Virtual Actor.

The Virtual Actor is composed of *Limbs*. Limbs are not to be seen as literal limbs of the user or an avatar, but are each a high-level and specific abstraction for one part of interaction. Each Limb represents one aspect of the user's behavior that is relevant for the application. For instance, a Virtual Actor may have a foot Limb that represents the position of the user in the VE. A hand Limb may represent what the user is holding or where she is grabbing.

All Limbs contain coordinate information, an orientation and a position. Limbs can also be extended to hold more information, as needed for their domain. The Virtual Actor defines a hierarchy of Limbs, arranged in a tree structure (a directed acyclic graph). This is used in the same way a scene graph works, accumulating the position and orientation of each Limb in the hierarchy. When a parent changes its position or orientation, its children will be changed accordingly. This structure has a number of benefits. In the case of building up a user representation, the dependence of body parts on one another is obvious. For handling input data, this can also be helpful, for instance gloves that deliver finger orientations can be placed under the hand to get the right world position.

3.2 Three Phase Interaction Processing

The Virtual Actor is the unifying and central component of the concept, but the actual creation of the interactions is performed in the other components of the system. The first phase of ACTIF's interaction processing is the acquisition of data from input devices and mapping of that data onto the Virtual Actor. This may be the abstract mapping of desktop input or direct mapping of a tracking data onto the Virtual Actor's Limbs. The next phase is to determine the intended interaction, by interpreting the actions of the Virtual Actor. Finally, that intention has to be executed in the environment.

3.2.1 Input Processing

The mapping of data from input device to the Virtual Actor is implemented in the *Device Mappers*. The main goal of this phase is to provide abstraction from the devices, both at a device interface level and at a semantic level. The Device Mappers perform their task in two phases: first polling the device and, then, mapping that data onto a specific Limb (or more than one Limb). The DeviceMapper class defines an abstract interface composed of two functions that implement the two phases of input processing: pollDevice() and updateActor(). All Device Mappers that are registered with the Kernel are run each frame.

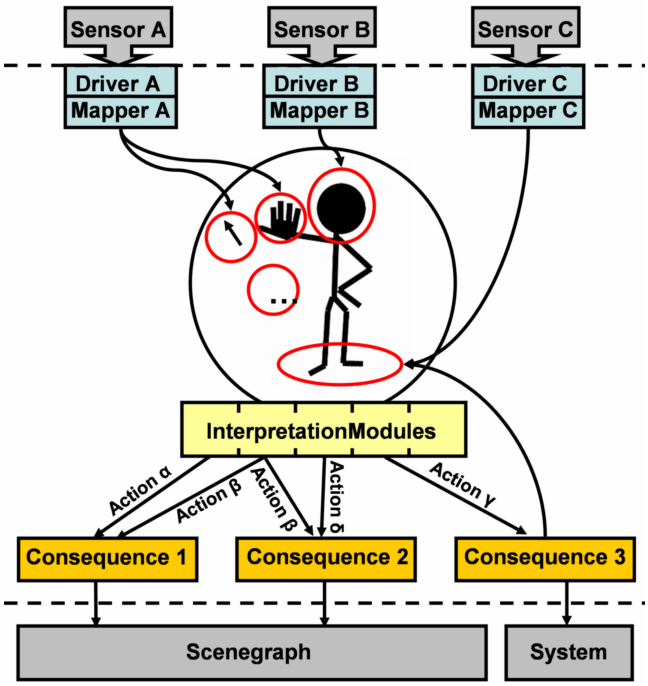

Figure 2: *The components of the ACTIF interaction frameworks implementation are highlighted in connection in this diagram.*

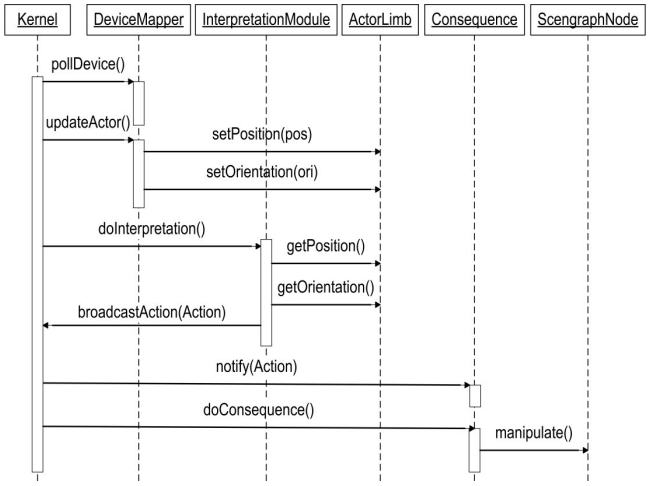

Figure 3: *The control loop of the ACTIF implementation is shown. Each of the Kernel calls shown is called for all active components of that type before moving on to the next stage.*

3.2.2 Interpretation

The next phase of processing is to react to changes on the Virtual Actor. The intentions of the interactor have to be interpreted from the data in the Virtual Actor and changes to them. This functionality is embedded into *Interpretation Modules*. They observe the interactor's behavior based on the data from the Virtual Actor and notify the framework when relevant actions should be performed. The Interpretation Modules are the components where the most of the interaction logic is found. The InterpretationModule contains a single function, doInterpretation(), that is called by the Kernel. Typically, an interaction technique consists of more than one Interpretation Module. A complex task is divided into basic subtasks. Each subtask is written as an Interpretation Module. This enables piecewise development of complex interactions and the modules can easily be reused in other interaction techniques as well.

Interpretation Modules do not affect the VE directly. Instead, the interactor's intentions are encapsulated in *Actions*. Actions are simple messages that indicate that a particular change to the system should be performed. The Action contains the necessary information from the interpretation phase so that the changes can be affected. They establish an extensible event system. Separating the detection of the interactor's intentions from their execution increases the reusability of the interpretations, as they are no longer specific to certain objects, etc. Through the loosely coupled event mechanism these components form, changing connections in new ways is simple as well as the creation of multiple reactions to a single event.

3.2.3 Consequences

The last phase of an interaction in the ACTIF system, is affecting the interaction on the virtual world. This phase is the abstraction layer to the specifics of the virtual world representation of the VR system. As such, it is highly dependent of the VR system. The working of this phase is implemented as *Consequences*. The registered Consequences are called after the interpretation phase to execute the Actions sent by the Interpretation Modules. Each Consequence reacts to specific Actions. changing the virtual world based on the information contained in those Actions. Multiple Consequences may react to the same Action and a single Consequence may react to multiple Actions.

3.3 Control Loop

The control loop of ACTIF is performed by a central mechanism, the Kernel. Although it is of fairly trivial complexity, it's necessary to understand its strict cycle. Figure 3 presents the control loop graphically. Each phase of processing is executed completely before continuing to the next phase. To enable systems of multiple interactions and modally phased interactions, a state system is implemented in the Kernel. The Kernel determines at the beginning of each frame the components that are in active states and executes only those components.

First, the method pollDevice of all DeviceMappers is called to retrieve new data via the device interfaces. Next, the updateActor method of the DeviceMapper's is called to propagate input data to the Virtual Actor. These calls are divided, so that the data of all inputs is collected as close to the same moment as possible. After the Virtual Actor is completely updated, the Interpretation Modules take their turn and the framework calls each modules doInterpretation method. Each Interpretation Module examines the Limbs of the Virtual Actor in which it is interested and determines if the changes in the Virtual Actor can be interpreted as certain Actions. If so, the Action message is sent using the Kernel's broadcastAction method.

Once all Interpretation Modules have been processed and, thus, all the Actions have been collected, the Kernel notifies all active Consequences of the Actions that exist. The doConsequence method of each Consequence is then called. An ordering of the Consequences is enabled by the addition of a priority level. Within each priority level, the ordering is considered random. If the Consequence receives the Action(s) necessary for it to run, it performs all changes in the underlying Virtual Environment. The Consequence may have side-effects within the framework. It may update portions of the Virtual Actor, or it may also cause state changes in the Kernel. Changes to the Virtual Actor will be seen by any Conse-

quences yet to be executed. The state changes take effect first in the next cycle.

3.4 Implementation of VR Systems Specifics

ACTIF is designed to localize the dependencies on systems specifics, making integration in different systems easier. There are three areas that require integration: the Device Mappers, the system Kernel, and the Consequences. The Device Mappers are dependent on the device or low-level interface that they use to retrieve the input. The ACTIF Kernel has to be integrated into the VR system's update loop. The centralized control of the Kernel means that this involves a single call in the update loop. Finally, the Consequences are dependent on the software used to represent the virtual world. This typically means adapting existing Consequences or writting them to deal with the appropriate scene graph.

ACTIF has been integrated into two different VR systems and uses a number of different input systems. The initial development of the system was done in the AVANGO system [Tramberend 1999]. This required the development of special structures in order to introduce it into the tightly controlled cycle of AVANGO. The Kernel is run as a service, so that it is first to execute. ACTIF is additionally embedded into the scripting language that AVANGO uses, making the setup of the interaction set easy. A port to VR Juggler was completed in short order [Bierbaum et al. 2001]. Bindings to both the OpenGL Performer scene graph and OpenSceneGraph (OSG) have been created.

4 Discussion

Our experience so far with ACTIF has been very positive. The basic concept fits well to the developer's needs when programming interactions. Creating new and experimental interactions is straightforward, since the descriptions of interactions can be followed in their implementation. This proved particularly helpful for novices to interaction creation. The division of the complete interaction into the three phases seems to be easy to grasp and to map into an implementation. The componentized nature of the system, in combination with the well defined interfaces, allows good reuse and composition of components to create new interactions. With a basic set of developed components, many interactions can be programmed by extending a single element.

The interworking of the Interpretation Modules and Consequences is one area that deserves consideration. The ideas was to move all interaction with the graphics system to the Consequences. For many interactions this is trivial to do. However, in some cases, the implementation of highly VE dependent interactions is complicated when adhering this strong separation. An alternative is to allow some knowledge of the graphics systems into Interpretation Modules. The unfortunate part of this is that the Interpretation Modules become system dependent. From a system side, there is little way to enforce the strict separation from the system in the Interpretation Modules. Although we have yet to find a case that cannot be relatively easily addressed with this strict separation, such cases may exist. We are continuing to look for mechanisms to make the implementation clearer in this regard.

A possible direction that would be interesting to look at in the future would be to change the implementation of portions of the system to leverage previously developed systems. In particular, the decision to make the Device Mapper a single component could be reconsidered. One possibility could be to simply separate the two components. An attractive approach for this phase would be to implement an openTracker-like filter system [Reitmayr and Schmalstieg 2001]. For trivial mappings, this is a bit of an overkill, since the current system works well. However, for the implementation of complex filters or in applications that require sensor fusion, the current method lacks support.

5 Summary

In this paper we have introduced a new framework for the development of interactions. The structuring of interaction support is approached from an interactor centric perspective. Building from this focus, the framework divides the interaction into three phases: mapping of devices to the Virtual Actor that abstractly represents the interactor, interpreting the Virtual Actor's state into actions, and, finally, changing the environment as the consequences of those actions. The interpretation and consequence phases are connected via an event based system, allowing flexibility and greater system independence. The state and priority mechanisms built into the control structures of the system support the design and implementation of complex interactions and environments containing a multitude of interactions.

An implementation of this interactor centric design has been presented ACTIF. ACTIF is designed to be adaptable to various VR systems, and implementations for two VR systems and three rendering systems already exist. The component based software design enables easy reusability and composability in interaction creation. ACTIF has proven flexible and an easily understood approach even for novices.

6 Acknowledgments

We would like to thank Gideon Otte for his assistance in the development of the Interaction Framework. We would also like to extend our thanks to the members of the VR course that implemented their interactions in a system still in development.

References

BIERBAUM, A., JUST, C., HARTLING, P., MEINERT, K., BAKER, A., AND CRUZ-NEIRA, C. 2001. VR Juggler: A Virtual Platform for Virtual Reality Application Development. In *Proceedings of the Virtual Reality 2001 conference (VR'01)*, 89.

FIGUEROA, P., GREEN, M., AND WATSON, B. 2001. A Framework for 3D Interaction Techniques. In *CAD/Graphics'2001 August 22-24*, International Academic Publishers, Kunming, China.

KESSLER, G. D. 1999. A Framework for Interactors in Immersive Virtual Environments. In *VR '99: Proceedings of the IEEE Virtual Reality*, IEEE Computer Society, Washington, DC, USA, 190.

REITMAYR, G., AND SCHMALSTIEG, D. 2001. An Open Software Architecture for Virtual Reality Interaction. In *ACM conference on Virtual Reality Software and Technology (VRST'01)*, ADM.

STRAUSS, P. S. 1993. IRIS Inventor, a 3D Graphics Toolkit. In *OOPSLA 93 Conference Proceedings*, vol. 28, ACM SIGPLAN, 192–200.

TRAMBEREND, H. 1999. Avango: A Distributed Virtual Reality Framework. In *Proceedings of IEEE Virtual Reality*, IEEE Society Press, 14–21.

WINGRAVE, C. A., AND BOWMAN, D. A. 2008. Tiered Developer-Centric Representations for 3D Interfaces: Concept-Oriented Design in Chasm. In *IEEE Virtual Reality Conference, 2008. VR '08.*, IEEE, 193–200.