Tools, Mediators, and Interaction Operators: A Concept for Interaction in Virtual Environments

Henrik Tramberend Frank Hasenbrink
Bernd Fröhlich
GMD - German National Research Center for Information Technology

Abstract

We present a flexible and powerful concept for the modeling of interaction in virtual environments. Our approach is based on three basic entities: tools, mediators, and interaction operators. Tools define the basic interaction mode and are the interface to the user. Mediators are the interface to the scene graph and are attached to nodes in the scene graph. Interaction operators implement the actual interaction functionality and use a tool and a mediator as parameters.

We realized this interaction concept in Avocado, our framework for developing virtual environment applications. We implemented a variety of tools, mediators, and interaction operators for different application prototypes and found our concept to be quite convenient to use.

1. Introduction

Interaction in immersive virtual environments is often quite primitive compared to interaction with desktop applications. In particular, there are no established standards for the interaction and manipulation of virtual objects, which is often a central task in many applications. For such tasks, we found a virtual tools based approach to work quite well. This might be mainly due to the fact that this approach resembles how we work in real life, where we simply employ different tools for different tasks.

This paper describes the ideas and concepts behind a virtual tools based approach implemented in Avocado [3], our virtual environment framework. Virtual tools can also be used to navigate in the environment, but here we focus on how virtual tools interact with virtual objects in a scene graph. We have structured our interaction concept into three different entities: virtual tools, mediators, and interaction operators. Virtual tools define the basic interaction mode and they are the interface from the real world into the virtual world. Mediators specify at which level in a scene graph the interaction should occur and they also provide an interface into the scene graph. The interaction operator implements the interaction functionality and it takes tools and mediators as operands. Interaction operators are temporary objects and they are instantiated for each interaction and discarded afterwards. This reflects directly the temporary nature of the interaction process.

From developing different application prototypes, we found that our interaction framework often lets us express our interaction ideas in a straight forward manner. Nevertheless, the virtual tools based approach described in this paper should be seen as only one interaction concept to choose from and it needs to mix and match with other interaction metaphors. We have used it for example in conjunction with passive real world props and found these concepts to work seamlessly together.

2. Avocado

Avocado is an object oriented programming framework for building distributed interactive virtual environment applications. Avocado is based on SGI Performer. It augments the Performer API with fields and field-connections, concepts well known from other graphics APIs like OpenInventor and VRML. Fields represent the state of objects. Field connections are used to build a data flow graph orthogonal to the scene graph. Avocado's field interface is the basic building block for the distribution. Multiple Avocado processes are kept synchronized by simply keeping the fields of shared objects in sync.

Avocado supports fast application prototyping and scripting through the interpreted language Scheme. All Avocado objects like nodes in the scene graph can be created and manipulated from Scheme at run time. Applications are often a collection of scripts written

in Scheme. These scripts instantiate Avocado objects, manipulate field values, and handle the data flow between them.

3. Basic Building Blocks

The development of our interaction concept was driven by the desire to meet the following set of goals:

- The specification of interaction possibilities should be orthogonal to the scene graph. We wanted to be able to add interaction to an existing scene graph without modification of the scene graph structure. This directly corresponds to the way our virtual environments are authored. First, the geometric structure of an environment is defined, and then the interaction possibilities are specified.

- The selection of different, abstract interaction modes should easily be possible. Most users are familiar with the concept of different interaction modes from their work with 2D user interfaces. Our approach should enable the application designer to use this concept for 3D user interfaces in order to present a familiar interaction environment to the user.

- The interaction concept should not depend on the selection mechanism used. In particular, our approach should support any mechanism that allows the selection of a single, specific object in a scene.

- Simple things should be simple! The specification of simple, every-day interaction patterns should be simple and straightforward, while the interaction concept should still be mighty enough to specify more complex interactions.

- The implementation of our interaction concept should lead to a set of general-purpose building blocks that support Avocado's scripting interface. This would enable the application programmer to quickly assemble new interaction patterns using only the scripting facilities. This can be done at run-time, leading to a very short development cycle for interactive applications.

To achive these goals we structure our aproach around three basic concepts: virtual tools, mediators, and interaction operators. The following sections describe each concept in more detail.

3.1. Virtual Tools

Virtual tools, similar to [1] and [2], define the basic interaction mode. They allow users to perform specific tasks like rotating, scaling, or changing the color of objects. Tools are typically attached to input devices, which contain 6 DOF sensors and other inputs from the real world like buttons or potentiometers. A typical input device could be a tracked wand equipped with buttons or a data glove with a 6 DOF sensor mounted on the back. Tools also carry additional parameters required for the task, for example a "color editing tool" knows about a color palette, a "drag tool" has a highlight color for high-lighting the object being dragged, and so on.

3.2. Mediators

Mediators are attached to nodes in the scene graph and specify at which level in the scene hierarchy an interaction may take place. Mediators also provide the interface to the scene graph through which an interaction occurs and they carry object specific interaction information. Mediators come in different flavors, e. g. a "matrix mediator" allows the specification of a matrix, a "material mediator" provides an interface for specifying material properties, a "script mediator" carries a Scheme script, and so on. Multiple mediators can be attached to a node to allow the manipulation of different properties.

3.3. Interaction Operators

Now that we have equipped the user with a virtual tool and the scene graph with mediators, we need to specify how and when a tool and a mediator interact with each other. Here we introduce the concept of interaction operators, which implement the actual interaction functionality and take tools and mediators as operands. The tool parameterizes the interaction from the user's side and the mediator parameterizes the interaction from the object's side. Interaction operators are temporary objects, which exist only for the duration of the actual interaction. This property nicely reflects the temporary nature of common interaction processes, e. g. picking up an object, moving it somewhere, and dropping it off.

A two-dimensional default interaction matrix maps compatible tool-mediator type pairs to interaction operators. For example, the interaction matrix might map the "drag tool" - "matrix mediator" pair to a "drag operator". The default interaction matrix is a subset of the tool-mediator compatibility matrix (Figure

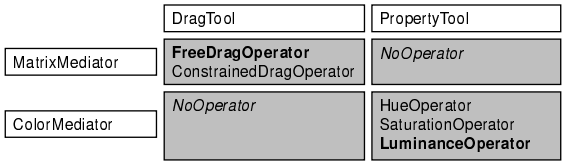

Figure 1. The tool-mediator compatibility matrix list the possible interaction operators for each tool-mediator combination. The default interaction matrix is a subset of the compatibility matrix. This example shows possible operators for a few selected tool-mediator combinations. The default interaction operators are set in boldface.

1), which lists all compatible interaction operators for tool-mediator pairs. Selecting an object in the scene starts an interaction process. Our selection mechanism is currently based on ray-object intersection, but it could be replaced by any other mechanism that selects a single object. By pressing a trigger button on the input device connected to the virtual tool, a ray is send into the scene. When it hits an object, the scene graph is traversed bottom-up searching for mediators. Whenever a mediator is found, it is checked if the interaction matrix contains an entry for the current tool-mediator pair. If that is the case the interaction operator is instantiated, otherwise the search continues.

The default interaction matrix is a convenient way to handle interactions in most cases. Sometimes this process needs to be refined for two different reasons: First, there might be different interaction operator implementations available for a certain tool-mediator pair. The interaction matrix can only provide a single default. Second, there might be multiple compatible mediators along the pick path. This becomes especially important when using scripted interactions. For example, a ”script mediator” carries a Scheme script that is executed when the user clicks on an object. Such a script could implement different interactions, one script might change the level-of-detail, another might iconize an object and so on. Since both mediators are of type ”script mediator” both are compatible with the same set of tools and the first one in the bottom-up search along the pick path would always be selected. We address the first problem by allowing tools and mediators to override the default interaction matrix on a per instance basis. The second problem can be solved by introducing non-unique names for mediators and tools. This allows a mediator to list the tools it wants to interact with and vice versa.

4. Conclusions and Future Work

We have presented a concept for object-based interaction in virtual environments that separates the input and intention of the user, the object selection mechanism and interface to the scene graph, and the interaction process into three different entities: virtual tools, mediators, and interaction operators. We have used this concept to implement a variety of prototype applications, including geo science visualization, steering of automotive crash simulations, and interior design. We found that most of our specific interaction requirements could be easily mapped onto the described framework. The virtual tools based approach was often intuitive and immediately understood by our users, in particular when working with workbench type environments.

As a next step we plan to support the concept of compound mediators. A compound mediator unites multiple mediators into a single one. This allows building complex mediators from simple ones without any programming.

We have developed our interaction concept also considering distributed interaction. The virtual tool is typically a local entity, whereas the mediator is distributed with the scene graph. The most interesting detail is that the interaction operator can be configured to be either local or distributed depending on various issues including computational requirements and the amount of data that needs to be transferred between tool, mediator, and interaction operator.

References

[1] L. D. Cutler, B. Fröhlich, and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. *1997 Symposium on Interactive 3D Graphics*, 1997.

[2] T. Poston and L. Serra. The virtual workbench: Dextrous VR. In *Virtual Reality Software and Technology (Proceedings of VRST'94, August 23-26, 1994, Singapore)*, pages 111–122, Singapore, Aug. 1994. World Scientific Publishing.

[3] H. Tramberend. Avocado: A distributed virtual reality framework. In *Proceedings IEEE Virtual Reality*, 1999.