Modeling Dynamic Aspects in Virtual Interactive Environments

Mathias Seitel[1], Vivek Vaidya[2], Raghu Kokku[2] and Rakesh Mullick[2]

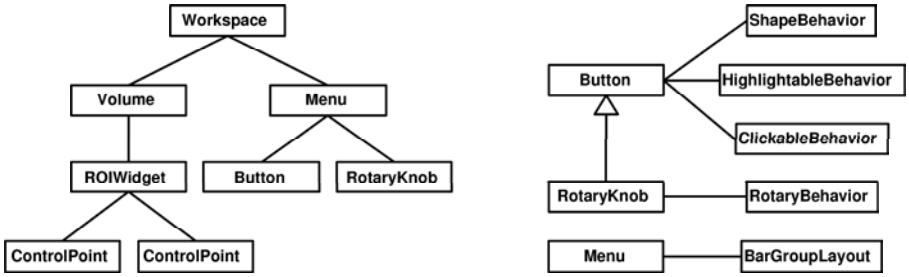
[1]Abteilung für Medizinische und Biologische Informatik,
Deutsches Krebsforschungszentrum, 69120 Heidelberg
[2]Imaging Technologies Lab, GE India Technology Centre,
Bangalore, India
Email: m.seitel@dkfz.de

Abstract. Virtual 3D environments are found in a wide range of applications, like entertainment software, industrial simulations and virtual reality in medical imaging. Modern graphics hardware supports real-time rendering of highly complex virtual worlds; at the same time, extensive interaction capabilities are required to make an application useful. Graphical scenes are commonly composed of many reusable and recurring elements in a well-defined hierarchical way. In this work, a similar approach is proposed for modeling interaction capabilities. Comparable to the elements of a graphical scene, interaction mechanisms often consist of multiple components. By modeling these components as reusable objects, complex interaction mechanisms can easily be realized. The usefulness of this method is demonstrated in an application for visualizing 3D medical data, in which it has been applied to design both the 3D user interface and several interactive tools for manipulating medical datasets.

1 Problem

Procedures for visually exploring three dimensional medical image data have become an integral part of clinical practice. While 2D slice views have proved to be most useful for precisely navigating through the data, 3D views usually offer a better overall picture. Many applications limit their interaction aspects (like data probing, defining regions of interests, or setting seed points) to 2D views and provide only basic navigation capabilities in 3D views. While this is reasonable for situations in which 3D would rather occlude details or hamper precise navigation, 3D still holds much untapped potential for interaction. Recent improvements in hardware (e.g. high-precision 3D input devices and graphics boards capable of dynamically visualizing 3D data) have further paved the way for new 3D interactive applications in medical imaging. One difficulty with 3D interactions is the increased level of complexity – creating 3D environments that appear natural and are intuitively usable is not trivial. Besides graphical components, the modeling of interaction aspects (what we call *Behaviors*) remains a major challenge.

Fig. 1. *InteractionObjects* and *Behaviors*. *InteractionObjects* can be composed to hierarchies and thus constitute the scene graph of the application (left). Each *InteractionObject* has a set of dynamically associated *Behaviors* which define its interactive capabilities (right). A *RotaryKnob* inherits all *Behaviors* of a general *Button* and adds a *RotaryBehavior* which allows the button to be grabbed and rotated around the Z axis. Events are issued at each rotary movement to inform consumers about the new rotation angle (which can be linked to a control value in the application). A *Menu* has an associated *Layout* which takes care of the spatial arrangement of the *Menu's* children.



2 Related Work

A large variety of systems for visualizing medical image data is available nowadays. Besides commercial applications, various scientific platforms and open source solutions exist, like the *Visualization Toolkit* (VTK), a popular framework offering many 2D and 3D visualization features and interaction capabilities. Systems like the *Medical Imaging Interaction Toolkit* (MITK) [1] or *Slicer* [2] integrate and extend VTK functionality.

Human-machine interaction in virtual environments is a broad research field affecting many scientific areas. Various approaches for modeling interaction aspects exist [3, 4, 5]. Gleicher [6] describes a system in which interaction characteristics are realized with numerical constraints. Similar to our approach, the method prevents undesired exposure of internal object details and allows for a combination of complementing interaction aspects.

3 Methods

In this work we propose a new method for modeling dynamic aspects (interactions or animations) in virtual environments. Instead of integrating dynamic aspects into the virtual objects they are applicable to, we model them as universal components that can be arbitrarily associated with objects. The process of modeling dynamic aspects thus becomes more general and well-defined, which facilitates the creation of highly flexible and reusable interaction mechanisms.

Our approach focuses on modeling *Behaviors* – or dynamic aspects – separate from the environment's virtual objects (called *InteractionObjects* in the

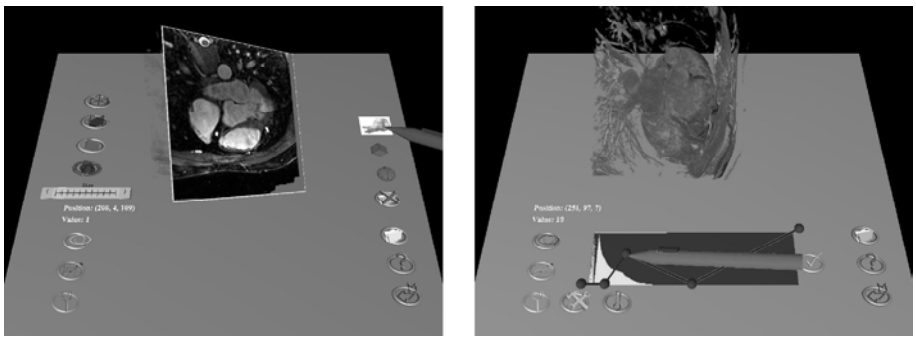
following). A *Behavior* can for example be the ability of an *InteractionObject* to be moved around by the user, or describe how an object is animated, or specify its reaction on mouse clicks. *Behaviors* are associated with *InteractionObjects*: A volume dataset may be grabbable (*GrabbableBehavior*), while a button may react on mouse clicks (*ClickableBehavior*). Any *InteractionObject* can be associated with a multitude of *Behaviors*, which allows for a flexible and fine-grained specification of its dynamic aspects. *InteractionObjects* have a visual representation and can form hierarchies; they make up the scene graph of the environment (Fig. 1). The main benefits of this concept are:

- Reusability: *Behaviors* are modeled independent of *InteractionObjects* and can be used in different contexts. Information exchange between *Behaviors* and *InteractionObjects* happens via events and abstract properties. Some *Behaviors* are dependent on others (for example an *AnimationBehavior* might only be activated when another *Behavior* detects that the user input device is hovering over an object). The logic of these dependencies can usually be encapsulated in the *Behaviors* themselves, using the concerned *InteractionObject* merely as a mediator.
- Dynamic *Behavior* changes: Due to the loose coupling of *InteractionObjects* and *Behaviors*, replacing the *Behaviors* at run-time becomes very simple. This can be particularly useful when different, context-dependent operations are to be executed on an object (e.g. probing at one time, window-leveling at another time), or for creating a highly dynamic 3D user interface, as will be exemplified below.

The concept of *Behaviors* and *InteractionObjects* facilitates the creation and management of complex interaction schemes. Examples for more sophisticated use-cases are:

- *Layouts*: In a user interface, objects often need to be grouped together, both logically and visually (for example buttons). In our approach, the arrangement of these objects is controlled by *Layouts*, specialized *Behaviors* that are associated with the parent object of a group. Each *Layout* implements a specific layout algorithm, determining the spatial arrangement of the objects - e.g. in horizontal or vertical bars, in a box, or in a circle. Since *Behaviors* are loosely associated with *InteractionObjects*, *Layouts* can be changed at run-time.
- *Animations*: Sometimes *InteractionObjects* are repositioned in the environment (e.g. when the *Layout* of their group changes). The *AnimationBehavior* serves the purpose of making this transition smooth. The object will be seen moving gradually from the old to the new location, which adds to the visual appeal of the application. Any *InteractionObject* can be made animated simply by including the *AnimationBehavior* in its list of *Behaviors*.
- *Tool specific Behaviors*: While some *Behaviors* (e.g. *Layouts*) are not directly associated with user-interactions, others relate to individual *Tools* (the virtual representation of input devices). An environment with different *Tools*

Fig. 2. The *VizDrive* application. The 3D user interface provides interaction capabilities for controlling the application and manipulating datasets. *Left:* A dynamic MR scout scan projected on top of a clipping plane. *Right:* Changing the volume rendering transfer function with a 3D widget.



can associate each *Tool* with a distinct set of interactive capabilities. The left hand *Tool* could be used to grab and hold a dataset, while the right hand *Tool* (possibly haptic enabled) would permit probing or manipulating the dataset. *Tools* can activate certain *Behaviors*, and *Behaviors* in turn affect *InteractionObjects*.

4 Results

The concept has been implemented in *VizDrive* [7], a system for stereoscopic visualization of medical image data with hand-immersed interaction support (Fig. 2). The application includes various components that are based on interaction mechanisms, for example probing instruments, ROI widgets, segmentation plug-ins, visualization controls, or the 3D user interface itself. In creating these components, our approach of modeling interactions has proved to be very effective. Assembling rather complex interaction patterns becomes straightforward as we can resort to a set of universal *Behaviors*. Transient object properties are easily realized and mainly avail the graphical user interface. All these factors lead to a clean and maintainable software design.

5 Discussion

We have presented a novel approach of modeling interaction aspects in virtual 3D environments. Our concept promotes a modular design of dynamic aspects in 3D environments. We believe that the proposed separation of dynamic and static aspects can significantly help in simplifying the creation of practical virtual tools.

Future work consists in further exploration of *Behavior* capabilities. Haptic aspects could for example be realized with special *Behaviors* that can query an

object's geometric features. By grouping interdependent *Behaviors* into specialized units, reusing standard object characteristics could be further simplified.

References

1. Wolf I, Vetter M, et al. The medical imaging interaction toolkit. Medical Image Analysis 2005;9(6):594–604.
2. Gering D, et al. An integrated visualization system for surgical planning and guidance using image fusion and an open MR. Journal of Magnetic Resonance Imaging 2001;13(6):967–975.
3. Kallmann M, Thalmann D. Modeling Objects for Interaction Tasks. In: Procs Eurographics Workshop on Animation and Simulation; 1998.
4. Döllner J, Hinrichs K. Object-Oriented 3D Modeling, Animation and Interaction. The Journal of Visualization and Computer Animation 1997;8(1):33–64.
5. Renzulli P, Kreylos O, Klein E, Staadt O, Hamann B. Toward Higher-Level Interaction Paradigms for Virtual Reality. In: NSF Collaborative Virtual Reality and Visualization Workshop; 2003.
6. Gleicher M. A Graphics Toolkit Based on Differential Constraints. In: Proc. ACM symposium on User interface software and technology; 1993. p. 109–120.
7. Mullick R, Hardy C, Raghu K, Darrow R. A Novel Hand-Immersed Paradigm for Interactive Medical Image Acquisition. In: Procs MMVR (Medicine Meets Virtual Reality). vol. 111; 2005.