A GENERIC USER INTERFACE FRAMEWORK FOR VIRTUAL REALITY APPLICATIONS

Frank Althoff[*], Thomas Volk[+], Gregor McGlaun[*], Manfred K. Lang[*]

[*]Institute for Human-Machine-Communication, Technical University of Munich, 80290 Munich, Germany
[+]Blaxxun Interactive, Elsenheimerstr. 61-63, 80687 Munich, Germany

althoff@ei.tum.de, thomas.volk@blaxxun.de, mcglaun@ei.tum.de, lang@ei.tum.de

ABSTRACT
Current VRML browser implementations lack the flexibility of adaptable user interfaces and navigation modes, which is in massive contradiction to the primary motivation for using 3D, namely to give the user a more natural understanding of how to interact with computer systems, and, in return, to make computer work more time-efficient. Multimodal interaction, such as the use of gesture and speech recognition, promises further improvement of the usability of 3D applications, but can only be seen in dedicated applications thus far. We present a user interface framework including the definition of a node set and the interfaces to other devices: As a key feature, we propose a *DeviceSensor* node that allows grabbing arbitrary user input, and a *Camera*-node to realize arbitrary navigation modes. The interface is based on the formalism of a context-free grammar providing the representation of domain- and device-independent multimodal information contents.

1. INTRODUCTION

The study of user interfaces (UIs) has long drawn significant attention as an independent research field, especially as a fundamental problem of current computer systems is that, due to their growing functionality, they are often complex to handle and therefore they require adaptation by the user to a high degree. Virtual reality (VR) interfaces (three dimensional UIs) resemble the latest step in the development of man-machine interfaces. Being highly interactive and immersive, they provide the most intuitive approach to communicate with the machine [Gre92], especially for non-skilled computer users.
Imposing enormous constraints on hard- and software, virtual reality evolved to a cutting-edge technology, integrating information-, telecommunication- and entertainment issues [Bul95]. Research in 3D user interfaces is a highly interdisciplinary task relying on computer and cognitive science, psychology and human factors analysis. VR aims at enabling the average user to intuitively communicate with information systems to easily realize and manipulate complex data content. Therefore, the applications of VR cover multiple domains ranging from 3D animation and computer games through scientific visualization up to complete virtual environments.

2. MOTIVATION AND TARGETS

Nowadays, VRML-based 3D applications are typically embedded in an HTML frameset since most applications also use HTML to show text-based content. Quite often, the user is overwhelmed by the range of functionality that a web page offers. Besides UI elements in the 3D window, the user has to cope with others in HTML, naturally these differ considerably from the look and feel of the 3D elements, not to mention all the elements of the desktop system, which leads to an overall decrease of usability. Usability tests underline that fact, revealing that even a 3D walkthrough could be an overkill for the end user. On the other side, VRML applications in the CAD / CAM domain and VRML-based games lack more elaborate navigation features. Neither can the built-in navigation features of the browser be adapted to the needs of the application, nor does the 3D author have any flexibility to create a unified UI.
From our experience with commercial 3D applications, we can derive the requirements of a UI toolkit to design customizable UIs and navigation modes. On the low end this toolkit has to deal with a primitive navigation driven by the cursor keys and switching on a light bulb by pressing another key; on the cutting edge, it has to cope with complex applications that involve several runtime modules themselves, make use of highly realistic 3D models, and are controlled by speech recognition systems and 6DOF input devices. The GUI of the browser functionality should be an integral part of the HTML frameset or of the 3D window. Desired features of the UI-toolkit are: (1) Allowing the implementation of arbitrary navigation modes with VRML, (2) Easy and intuitive to use, no need to bother 3D math into depth, (3) Support for all kind of input devices and (4) Enabling the design of arbitrary user interfaces.

From: Proc. HCI 2001, Lawrence Erlbaum Ass., NJ

2.1 VRML Navigation Features and Event Handling

Regarding the event handling of a browser implementation we will find a rigid event routing between the input events like mouse movements and key strokes and the input handling, that realizes the navigation, other UI elements and the VRML sensors, e.g. a mouse drag over a anchor node is handled by the anchor node, outside the anchor the movement serves as the input for the navigation module. The processed events of the sensor nodes fire in return events connected to event consumer fields by a ROUTE mechanism. The author has no influence on the event flow and the event handling. Neither can he change the reaction of the navigation module on user interaction nor he can grab events in VRML directly for implementing a 3D UI.

The navigation module exposes only a small set of its functionality by the *Viewpoint* node. Various viewpoints can be defined and activated by binding the viewpoint to the browser. Wrapping viewpoints in *Transform* nodes and applying animations to them, guided tours can be implemented. Even some simple navigation modes can be implemented by catching user input with *TouchSensors,* using them as input for computing a new user position and orientation in a *Script* node. As a drawback, no collision detection is performed and a lot of 3D math is involved.

In the framework described in this paper we use the mechanism of ROUTEs to break up the rigid event routing. Additional devices can plug in the event dispatcher module of the browser and dispatch their raw input to a *DeviceSensor* node in the VRML scene graph. In addition, a device can trigger certain browser actions by using some formalism explained in the next section. At VRML scope, the newly defined *DeviceSensor* node gives access to all kinds of conceivable user input. The *Camera* node gives superb control over the camera used for rendering: By setting velocity vectors, the VRML author animates the camera. The camera computes the distance vector for the current frame in consideration of previous frame times and the results of the collision detection. Besides a 6DOF navigation mode the node supports an EXAMINE and a BEAMTO mode, thus covering all conceivable navigation modes while hiding the nasty details of 3D math from the VRML content author. However, in the absence of a *Camera,* a default built-in navigation should encompass a minimal navigation feature set allowing all basic navigation actions such as the WALK, SLIDE, EXAMINE modes.

As a consequence of the flexible event handling any given VRML geometry can be used to implement a user interface. World-specific navigation panels and menus can be an integral part of the scene. A panel can easily be set up with HUDs incorporating *TouchSensor* nodes. Menus can configure the browser by dedicated vrml-script calls to the browser and even drive other applications by using *eventOut*-observers of the EAI.

2.2 Interface to Advanced Input Devices

Human beings are able to process several interfering perceptions at a high level of abstraction so that they can meet the demands of the prevailing situation. Most of today's technical systems are incapable of emulating this ability yet, although the information processing of a human being works at a plainly lower throughput than can be reached in modern network architectures. Therefore, many researchers propagate multimodal system interfaces [Eng98], as they provide the user with more naturalness and flexibility, and also work in an error-robust way.

In our work, we present an approach for handling multimodal information in a VRML browser that can be generated by arbitrary and also multiple input devices. Thus, advanced input devices like data gloves, motion capture systems or even higher-level components like speech and gesture recognition modules can be plugged into the event dispatcher in addition to the standard control devices like keyboard and mouse. For interfacing to the browser we are using an abstract communication formalism based on a context-free grammar (CFG).

Based on this abstract model we are able to represent domain- and device-independent multimodal information contents. Thus for example, both natural speech utterances and hand gestures are described in the same formalism. As the individual input devices all share the same formalism, it makes no difference to the browser module by which exact input device a specific event has been generated. The browser module just operates on the formal model of the grammar. An additional advantage of this approach is that arbitrary new information sources can easily be integrated into the interface, too.

Messages and events created by the various low- and high-level devices from simple keystrokes to complex natural speech utterances are semantically unified and mapped on the grammar formalism. Based on the created information contents, the CFG-module triggers the specific browser functionality by using the EAI. For navigation, this mainly concentrates on routing the commands to the camera node described above. The concept has proven to be working in a prototypical implementation which can be adapted by the user according to his individual needs.

The key feature of our approach using the CFG-modul as a meta-device interfacing the various input devices to the browser is that it provides a high level access to the functionality of the individual recognition modules. By changing the underlying context-free grammar the interaction behavior of the target application can easily be modified also by non-experts without having to deal with the technical specifications of the input devices. The commands of the grammar are comprehensible straight forward as they inherently make sense.

3. EXTENSION NODES FOR CUSTOMIZABLE INPUT HANDLING AND NAVIGATION

In the following, we will introduce a Camera node to realize arbitrary navigation modes and a DeviceSensor node that allows grabbing arbitrary user input. Implementation details on these proposals will be given elsewhere.

3.1 Camera Node

We propose a new Camera-node, providing the three basic modes SIXDOF, EXAMINE, and BEAMTO. All standard modes like WALK, SLIDE, PAN, FLY are regarded as a subset of the SIXDOF mode. In the EXAMINE mode, the camera is moved on a virtual sphere with a specified center and radius. The camera target is not changed, i.e. the viewer moves on the virtual sphere, but does not look to the center automatically. In the BEAMTO mode, the viewer position is animated to a specified position, also turning the viewing direction to a new point.

The *Camera* node is derived from the *Viewpoint* node and inherits all *Viewpoint* fields, since its basic functionality is to control the camera, too. Like the *Viewpoint*-node, the *Camera* is bindable and is queued in the same stack as the viewpoints. To activate a certain *Camera,* it has to be in front of all other cameras and viewpoint definitions at file scope or has to be bound explicitly. Of course, the bind mechanism can be used to switch between various navigation modes if corresponding configured cameras are present in the scene.

3.2 DeviceSensor

Furthermore, we introduce a *DeviceSensor*-node that is capable of observing arbitrary input devices such as a 6DOF mouse or a speech recognition system. The device data is wrapped in an *Event* node that already covers a considerable amount of possible event types. Special-purpose devices can replace the default implementation with their own *Event* node, guaranteeing maximum flexibility for the support of all input devices.

A *device* field specifies the hardware device that is observed by the node. A sub-device string allows further refinement of the selection of the device output data. Additionally, an ***eventType*** field determines which event types are to be observed. The ***eventType*** accepts multiple type values that are specified in W3C-DOM-Level2 [Do99]. For devices not mentioned here a new device name can be created. The node uses this string to identify the device driver plugged into the event dispatcher.

4. CASE STUDIES

To demonstrate the benefit of customizable navigation modes two reference implementations taking advantage of the *Camera* and the *DeviceSensor*-node have been made. The formerly used navigation panel and right-click menu is replaced by VRML-based implementations. With VRML-script calls the VRML-author can get the current status of the browser, e.g. status of texture smoothing or texture dithering, and set new values.

A very intuitive method of navigating in 3D is to animate the camera to the point that has been selected by a mouse click. As inexperienced 3D users are overwhelmed by “drag navigation” they try to reach their target by clicking on it. By using the selection test of the browser triggered by a mouse click of the user, the target point is determined. This point or a point close to it is used as the target and orientation point of a *Camera.*

Experienced users prefer to explore their 3D environment, using various navigation modes acting with the mouse and the keyboard simultaneously. Moreover, they demand additional browser features like the display of a third person view and having the possibility to interact with others in a multiuser environment. In our sample we apply the cursor-keys, allowing the user to look left and right, up and down while walking without switching modes.

5. CONCLUSIONS

By tailoring applications to user profiles as shown in our case studies we can enable a broader use of VRML applications. The proof of concept is already given by the implementation of the nodes in blaxxun contact 5.0 and several sample applications. Therefore, we propose to add the new nodes to the VRML200x standard. Working with the *DeviceSensor* and advanced input devices we came to the result that even further extensions are to be evaluated.

REFERENCES

[Gre92] Green, M. et al (‘92): Virtual Reality and Highly Interactive Three Dimensional User Interfaces. ACM Conference on Human Factors in Computing Systems, Mai 92 Monterey, ACM Press

[Bul95] Bullinger, H-J. et al (‘95): Virtual Reality as a Focal Point between New Media and Telecommunication. VR World ´95 – Conference Documentation, IDG 1995

[Do99] Document Object Model (DOM) Level 2 Specification (1999), homepage http://www.w3.org/TR/1999/WD-DOM-Level-2-19990304/events.html

[Eng98] Engelmeier, K.-H. et al. (1998): Virtual reality and multimedia human-computer interaction in medicine. IEEE Workshop on Multimedia Signal Processing, Los Angeles, Dec. 1998