Eurographics Symposium on Virtual Environments (2004)
S. Coquillart, M. Göbel (Editors)

Live Tuning of Virtual Environments: The *VR-Tuner*

Stefan Conrad

Hans Peter Krüger

Matthias Haringer

Fraunhofer Institut für Medienkommunikation, Sankt Augustin, Germany

Abstract

This paper describes a solution for the modification of virtual environment (VE) applications while being immersed in the application scenario inside of an immersive projection environment. We propose an infrastructure which enables developers to adjust object properties and change the structure of the scene graph and data flow between nodes using a tablet PC. The interface consists of a two dimensional graphical user interface (2D GUI) brought on a spacial aware touch screen computer, accompanied by a mixer console with motor faders. We discuss the usability of the combination of different interaction modalities for the task of tuning of VE applications.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: H.5.2 [User Interfaces]: D.1.7 [Visual Programming]:

1. Introduction

The production of applications for virtual environments requires a set of different skills and professions. Graphic designers, 3D modellers and programmers are working together to produce a scenario in the core work flows of modelling, application programming and VR-system programming [CKS*03]. The efficiency of the whole development process depends on the synchronisation of the work flows and successful composition of their artifacts in each project iteration.

Creative professionals are usually dependent on programmers and modellers to transfer their ideas and apply changes during project iterations. This can lead to enormous delays in the development process. A worst case scenario would be, that during a test run right before a presentation on a certain projection environment, it turns out that the lighting or position of some objects have to be adjusted or the chosen navigation metaphor does not suite the setup of the specific display. Todays state of the art in VE content production would require the skill of an expert in programming or modelling to overcome this problem by changing the application or the scene geometry source itself on a desktop system while the results have to be controlled in the IPE.

The *VR-Tuner* proposes a solution to this problem by offering a tool which allows users to change an already set up application scenario without having to interfere with the programming. It is part of the *alVRed* project (funded by BMBF, the German ministry for education and research), which aims to provide a set of tools for the creation of non-linear storytelling content for VE [WGT*02]. Though the *Tuners* initial design was inspired by the needs of storytelling content providers its use is not leashed to this. We would like to broaden the view here from story telling to scenario based applications in general. In the following we will show, how the *VR-Tuner* enables developers as well as non VE and programming experts to adjust a VE applications from within the virtual environment.

2. Related Work

Systems like *VEDA* [Ste96] and *dVISE* [div94] allow the definition of interaction processing (*VEDA*) and manipulation of symbolic object properties (*dVISE*) directly in 3D. The programming interface is part of the VE scene. This is a promising approach, especially for setups with head mounted devices where real computer hardware is unavailable for interaction. The flip side of this principle is the fact, that it ties the operating tool too tight to the operated subject. If the property of an object is for example causing the simulation frame rate to drop down to 0.1 frames per second, it will be hardly possible to use 3D interaction techniques to change this property within that environment. Since we were looking for a robust solution for use under real development conditions, we did not take this approach into account.

It seems to be more promising to decouple the programming tool from the VE itself as far as possible, without banning it on a desktop computer outside of the immersive projection environment (IPE). Interactive 2D GUIs which are used for VE development are known from various virtual reality (VR) development systems like *Virtools* [vir04] which deals with behavioural modelling, *xDVISE* which provides a view on the underlying VR systems scene graph and object properties as well as *EON Reality*'s *EON Studio* [eon04] which also allows interactive scenario creation based on visually wired building blocks. These approaches have in common, that they are dedicated to desktop use. Presenting these interfaces on a touch screen computer inside of an immersive projection environment (IPE) does not solve the problem. Their design and structure do not allow full operation in the absence of a mouse and keyboard.

The use of wearable touch screen computing devices inside of an IPE is documented by Watsen et al. [WDC99]. L. C. Hill investigated the general usability of PDAs (Personal Digital Assistants) and tablet PCs inside of IPEs [Hil00]. *Tweek* is a software architecture for the use of a PDA in a projection environment [HBCN02]. These approaches deal with GUIs for application control rather than for application development as it is demanded for the *VR-Tuner*. The major difference between application control and application development is the higher amount of symbolic data which has to be considered when doing the latter. Development sessions inside of a VE also require methods for finding objects with no or a non visible geometric representation like abstract data processing scene graph nodes or sound sources.

3. Developing Scenario Based VE Applications

The term *scenario based* denotes here all kinds of applications which rely on a scenery of well defined geometrical objects which is explorable using moving metaphors like walking or flying. Typical examples of scenario based applications come from the field of story telling, architectural design and industrial engineering, design review and assembly simulation. The exploration of a volumetric medical data set would not fall into this category. The core of a VE application framework is typically a scene graph based graphical renderer together with a set of device drivers for interaction devices and support for display of other sensual representations like sound and haptics. Support of scripting for fast prototyping is another feature of a mature VE system. Some systems enforce also the definition of data flow networks on the scene graph nodes according to routings between VRML (Virtual Reality Modelling Language) fields. *AVANGO* [Tra99] is a representant of a VR framework which supports scripting as well as the data flow programming paradigm based on scene graph nodes with fields and connections between these fields. The core work flows in the production of scenario based applications in such kind of systems are authoring (in the case of a storytelling application), modelling, application programming and VR-system programming. In the case of *AVANGO*, application programming means writing scripts in *Scheme* where the scene is constructed, nodes are instantiated, routings between fields are defined and callbacks to the scripting system are installed.Scripting and data flow based programming with visual tools as mentioned above are powerful concepts for rapid prototyping. When it comes to a later phase in the development process, when the code base of the application is grown and the scene is complex, these approaches show weak points in supporting further small development and testing iterations.

4. Design and Implementation of the *Tuner*

The *Tuner* follows the idea of presenting the interns of a running VE application to the person who is immersed in an IPE. For the presentation of the huge amount of symbolic data the use of a wearable touch screen computer is the first choice. Tests with a PDA showed that the screen is too small to perform the necessary tasks with an acceptable amount of context switches. We were using a *PaceBlade* and an *intermec* tablet computer to run a 2D GUI which is optimised for touch screen use. The *PaceBlade* (see fig.1,left) has a bigger screen with higher resolution (1024*800). The screen was big enough to realise touch screen interaction with bare fingers without pen for most tasks. The *intermec* (Fig.1,right) has a resolution of 800*600 on a smaller screen, but it is much lighter and provides numeric and cursor keys. In a further step during the design of the *Tuner* design, we combined the touch screen with a standard MIDI (Musical Instrument Digital Interface) mixer console with motor faders (see fig. 2). The touch screen can be attached to the mixer in a dedicated case and then be used stationary (see fig. 3).

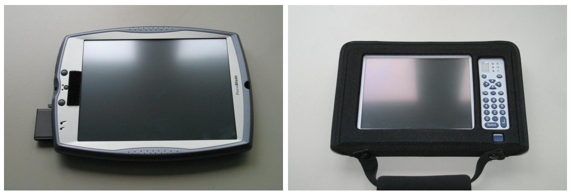


Figure 1: *Two touch screen computers which are used for tuning.*

Figure 4 shows the basic software infrastructure of the *Tuner* interface. The VR system exposes its internal state to the outer world through the RUI (Remote User Interface) Server. The RUI protocol allows clients to get and set node and field states, and to retrieve and change the structure of the scene graph and the data flow network defined by field

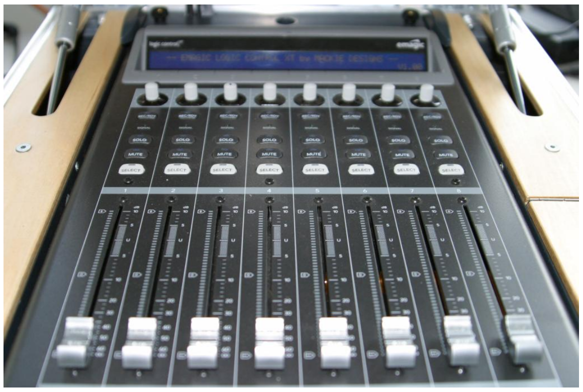


Figure 2: *A MIDI mixer with motor faders for hands on control of node properties.*

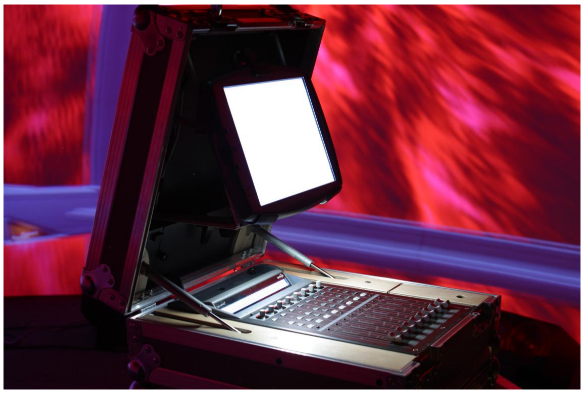


Figure 3: *The touch screen attached to the MIDI mixer in the* iCone *VE display.*

connections. It also provides meta information about the fields like default value and min/max range for numerical values. These operations work independently from the render loop, to avoid clients to be blocked by low frame rates. The RUI client is running on the touch screen device with an 11Mbit wireless LAN card and connects to the server via TCP/IP sockets. Server as well as client are implemented in C++ and use low level socket communication to gain the best performance on the wearable computer. The client provides a binding to a 2D GUI toolkit (Qt) and a MIDI binding to support the optional use of a standard MIDI mixer console with motor faders to control field values. The RUI client subscribes as an observer to specific nodes in the VR system. The RUI propagates changes of field values directly to elements of the 2D GUI and motor faders and back and thus provides a continously updated view on the VR-System.

Figure 5 shows a snapshot of the GUI with a browser for the scene graph nodes on the left side, showing the nodes of the current scene which the user is in. In the default mode

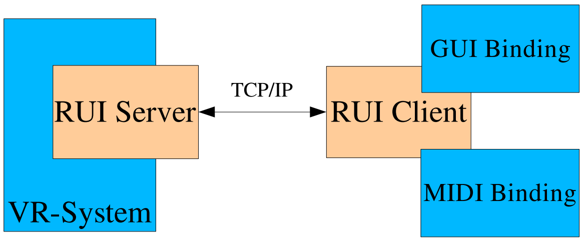


Figure 4: *The software architecture of the Tuner interface.*

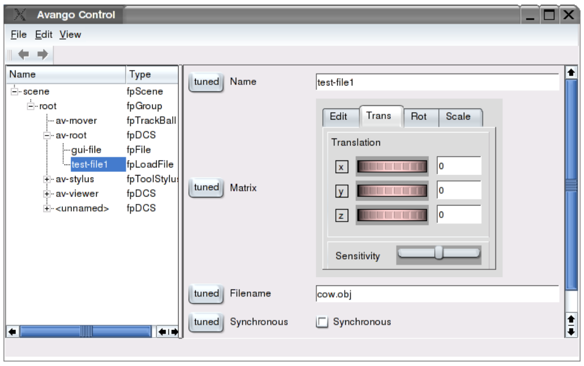
the list is sorted by object classes. In a second mode the objects are listed in a hierarchical tree view, according to their arrangement in the scene graph tree. The tree view can be given in a file browser like style or as picture of the directed acyclic graph. On the right hand side of the main window, the fields of the currently selected node are listed with their name and a dedicated control element for each value and type. They can set field values and receive immediate update information about field changes. The observer pattern guarantees a consistent interface across the GUI, motor faders and the VE system itself. The third main part of the GUI is a view on the data flow network built by the field connections presented as a directed graph. A 2D map view on the environment is used for navigation and selection of objects. The graph views on the scene graph hierarchy and the data flow network can be used to change the scene graph hierarchy and remove or establish field connections.

In practice the user selects a scene graph node via the scene graph browser, the 2D map or by 3D selection techniques. The fields of the node will be listed on the touch screen and the motor faders will move to the positions according to the values of the assigned fields. Now the user can change parameters with the faders or the GUI controls. At the end of a tuning session the set of changes are made persistent in an XML based file format. This tuner file is loaded on the next start of the application after the scripts for scene setup have been executed and restores the application to the tuned state. Different sets of tunings can be stored and retrieved on demand.

The *Tuner* provides a non expert mode for users without dedicated VR system programming. This enables creative members of the development team to adjust object properties without the indirection via a core developer who has to work on the source code. In this non expert mode only certain node and field types are visible and tuneable. The graph editors for the scene hierarchy and the data flow are not available. Following variables turned out to be useful as subject of the tuning process for non experts:

Transformation of objects in space: Translation, scaling and rotation can be adjusted.

Visual material properties of objects: Material properties as they are supported by the renderer.



Name	Type
scene	fpScene
root	fpGroup
av-mover	fpTrackBall
av-root	fpDCS
gui-file	fpFile
test-file1	fpLoadFile
av-stylus	fpToolStylu
av-viewer	fpDCS
<unnamed>	fpDCS

Name: test-file1
Matrix: x 0, y 0, z 0
Filename: cow.obj
Synchronous: ☐ Synchronous

Figure 5: *A screenshot of the* VR-Tuner *GUI.*

Light colour, position, orientation and intensity

Sound source properties: *AVANGO* offers sound sources as nodes which can be arranged in the scene graph and have position and direction, volume and filter parameters.

Navigation properties: Selection and parametrisation of the navigation model like flying, walking, viewpoint height of walker.

An important issue is the awareness of the user to what extent he is changing the environment from its original state. With the described interface a user can easily change a lot of critical properties with a few moves on the touch screen or the mixer console. On the level of fields, a toggle button for each field indicates whether this field has already been tuned manually. The tuning for a particular field can by reset to its initial value by pressing this button. The toggle button has its counterpart in a hardware button on the mixer console. The scene graph and data flow view indicates tuned nodes by a dedicated colour. The RUI server detects the overwriting of tuned values from script callback code as a conflict. This is also indicated by colouring the conflicting field or node in the GUI.

5. Practical evaluation

The *Tuner* has been used in three different projects and 9 developers have been reporting about their experiences with the setup. In a first evaluation round we used the touch screen together with a 3D stylus and the motor faders in a cylindric display. The styles was used for pointing in 3D and as a pen on the touch screen. Feedback from the developers lead to the following major results:

Combination of stylus and touch screen in hand: The use of the stylus for pointing as well as for pen interaction on the touch screen was not as useful as expected. Pointing in 3D and using GUI controls with the stylus tip requires two different hand positions (see fig. 6). Switching between these positions takes time and disturbs the flow of actions.

Combination of touch screen and mixing console: Hill discovered in his user study [Hil00] two needs of users who are using a PDA inside of a virtual environment which we could reproduce in our tests. Users wanted to be able to change continuous values with sliders *blindly*, without having to look at them. With GUI sliders the user has to switch its focus between the GUI to trigger the slider and the virtual environment to see the result. Another issue was the possibility to put the touch screen device down from time to time. The mixer console addresses these needs. The motor faders provide tactile feedback of the position and progress while moving them and can be used without switching the eye focus between controllers and environment. This was experienced as very positive by all users. Since the touch screen computer was also accessible without a dedicated pen, switching between mixer controls and GUI elements did not require a big change in hand posture. For the switching between mixer and GUI it was irrelevant whether the user was using the touch screen stationary attached to the mixer or standing next to the mixer, holding the touch screen in one hand. The biggest drawback of this combination is the absence of 3D interaction possibilities since a simultaneous use of a 3D interaction device with touch screen and mixer is not practicable. For 3D interaction a dedicated device has to be taken in hand, used separately and put back in place after the task has been performed. Experienced developers who did most selection tasks with the scene graph browser considered this lack as less severe.

Usability of the GUI on the touch screen: Some GUI interactions use the metaphor of mouse dragging (clicking on an item and moving the mouse with the button pressed). An example is the drawing of field connections in the data flow diagram. On the touch screen this is implemented by touching the surface of the screen and moving the pen or hand while keeping the pressure on the screen. Dragging on the touch screen requires continuous attention for the screen and was experienced disturbing inside of the virtual environment.

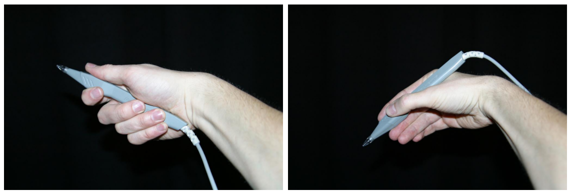


Figure 6: *Two different hand positions for pointing in 3D (left) and for use on the touch screen surface (right).*

After this evaluation we translated the experiences in changes to the system. The GUI was optimised for touch screen use by decomposing actions which required dragging before, in several simple pointing actions. A field connection for example, which was drawn between fields in the data flow diagram before is now established by selecting the source and destination field separately and confirming the connection by pointing on a 'connect' button.

As a main improvement we attached a tracker sensor to the touch screen device. By doing this we overcome the problems of the simultaneous use of touch screen and dedicated 3D interaction devices. The touch screen computer can now be used for 3D interaction tasks directly. The most important 3D interactions during tuning are selection and navigation. For selection, the touch screen device is sending a virtual pick ray as known from stylus use. It can be moved around like a *Tricorder*, showing the properties of objects, intersecting the pick ray. Because of its low weight and a fixing strap on the back of the device, the *intermec* was more useful for this kind of interaction than the bigger *PaceBlade*. Map based navigation and selection was improved by aligning the map with the virtual environment, independent from the current spatial orientation of the touch screen computer. Direct navigation is performed by using the current direction of the of the wearable computer together with hardware buttons on the *intermec* and software buttons on the screen of the *PaceBlade*. The user points into the moving direction and presses the buttons for forward or backward movement. This technique is applied for walking as well as for flying navigation models.

This setup is currently used in project development. The use of the touch screen device for direct 3D selection was evaluated as useful by the developers. Direct navigation with the touch screen was capable of replacing other temporarily not available navigation techniques (flight stick) for the duration of the tuning process. Its general usability has to be investigated in more detailed tests.

6. Conclusion and Future Work

With the *VR-Tuner* we have shown an instrument which enables users to adapt VE applications interactively. The use of a touch screen computer and hardware faders inside of the immersive projection brings WYSIWYG (What You See Is What You Get) to VE application development. The spatial aware wearable computer enables the user to perform 3D interactions during the tuning process without having to switch to other dedicated interaction devices. The next step will be, to use the *VR-Tuner* already in earlier steps of the development process as a main development tool. Rather than tuning an already programmed application it is also desirable to use the *Tuner* concept for primary programming. An in depth evaluation of the described interaction techniques is planned to explore their usability for general tasks beside the *Tuner* application.

References

[CKS*03] CONRAD S., KRUJIFF E., SUTTROP M., HASENBRINK F., LECHNER A.: A storytelling concept for digital heritage exchange in virtual environments. In *Proceedings of the 2nd International Conference on Virtual Storytelling, ICVS2003* (Toulouse, nov 2003).

[div94] DIVISION LTD.: *dVISE User Guide*. 19 Apex Court, Woodlands, Almondsbury, Bristol, NS12 4JT, UK, 1994.

[eon04] Eon reality website. http://www.eonreality.com, feb 2004.

[HBCN02] HARTLING P., BIERBAUM A., CRUZ-NEIRA C.: Tweek: Merging 2d and 3d interaction in immersive environments. In *6th World Multiconference on Systemics, Cybernetics, and Informatics* (jul 2002).

[Hil00] HILL L. C.: *Usability of 2D Palmtop Interaction Device in Immersive Virtual Environments*. Master's thesis, Iowa State University, Ames, Iowa, USA, 2000.

[Ste96] STEED A.: *Defining Interaction within Immersive Virtual Environments*. PhD thesis, University of London, 1996.

[Tra99] TRAMBEREND H.: Avango: A distributed virtual reality framework. In *Proceedings of the IEEE VR 1999* (1999).

[vir04] Virtools website. http://www.virtools.com, feb 2004.

[WDC99] WATSEN K., DARKEN R., CAPPS M.: A handheld computer as an interaction device to a virtual environment. In *Third International Workshop on Immersive Projection Technology (IPT)* (Stuttgart, Germany, 1999).

[WGT*02] WAGES R., GRÜTZMACHER B., TROGEMANN G., MOSTAFAWY S., JAIN R., HASENBRINK F., CONRAD S.: Nichtlineare Dramaturgie in VR-Umgebungen. In *Tagungsband der 2. Internationalen Statustagung 'Virtuelle und Erweiterte Realität' des BMBF* (Leipzig, Germany, nov 2002). german.