Interacting with Simulation Data in an Immersive Environment

Christian Knöpfle

Department for Visualization & Virtual Reality
Fraunhofer Institute for Computer Graphics (IGD)
64283 Darmstadt, Germany
Christian.Knoepfle@igd.fhg.de
http://www.igd.fhg.de/www/igd-a4/

Abstract. In today's automotive industry there is an increasing demand for VR technology, because it provides the possibility to switch from cost and time intensiv physical mock up's (PMU) to digital mock up's (DMU). Furthermore the visualization and examination of simulation results is a very important aspect during the whole development cycle. Therefore tools are needed, which enable the users to work with DMU's, as well as simulation data sets in an efficient and intuitive way. In this paper we present the design of a VR user interface for evaluating simulation data sets. The design of the user interface is based on the basic interaction tasks (BIT's), introduced by Foley et. al.. This allows to generalize the results presented herein and to apply them to other domains.

1 Introduction

Today's available computing power allows to simulate more and more aspects of the whole development cycle and consequently there is an increasing demand for tools, which are able to visualize these simulation results and allow to examine them. Most simulation packages provide viewers, but they are only capable to load a limited set of file formats and are operated in different ways. Since various simulation packages are currently in use, a single tool is needed, which is capable of loading data sets coming from different sources. Thus, the users have to master only a single toolkit.

The examination of simulation data is part of the design review process of upcoming products. During a design review, several experts discuss and investigate the simulation. They try to identify design flaws and to find solutions for them. The outcome of every design review session is a protocol, where the results of the meeting are written down. This protocol is handed over to the responsible person, to fix the problems. Therefore, a tool for interacting with simulation data has to fulfill the basic requirements for design review applications too. Analyzing today's procedure of design reviews, the basic requirements to a system, can be defined as follows:

1. easy to use and fast to learn interface

2. various functions for investigating the model, like clipping, changing visibility and measuring
3. ability to create screenshots, placing markers and adding text for documentation purposes. The documentation should be saved as HTML. Then the documentation can be put in the Intranet, or PDM system, where the responsible person can easily access it without starting a specific tool

Using Virtual Reality techniques to build such a system seems to be a very promising way, because VR offers specialized input and output devices, real-time rendering of even very complex datasets and the possibility to build intuitive and natural user interfaces.

Because of space constraints, we will mainly focus on the interaction with simulation data. Since the proposed concepts are based on the basic interaction tasks and are not tied to the functionalities itself, they can be easily adopted to other types of design review applications.

Next we will describe the functional requirements for the interaction with simulation data. Then we discuss the hardware setup, which may influence the design of the user interface. Afterwards the basic interaction techniques are introduced and their realization described. Finally the prototype is presented.

The development of the whole system was done in cooperation with the BMW AG and based on the users input enhanced and improved.

2 The Basic Requirements

For the visualization of simulation data sets, the following requirements can be defined:

1. Visualization of the simulation data
2. Translate and orient the whole model in an intuitve way
3. Show and hide of different groups
4. Playing back the different time steps of the data set. The direction and speed should be defined by the user
5. Data probe and display of the according data value
6. Display the time dependant values of a data probe as a curve
7. Changing the color-mapping interactively
8. Documentation

We focus on the visualization of 2D data in 3D space (surface model) with a 1D value per node, encoded as RGB color. Nevertheless, the presented metaphors for interacting with simulation data can be easily adapted to any other type of data.

3 Input and Output Devices

Each application area has its own requirements, which influence the software and the hardware interfaces. In this chapter, we will examine briefly the demands of a design review application to the hardware interfaces.

An input device is needed for the interaction with the virtual scene. Given a permanent installation used for design review, which will be shared by several working groups and used on a daily basis, the operability has to be guaranteed and therefore robust devices are required. During a session, these devices have to be easily handed over from one person to another, giving every participant the opportunity to operate the system. Furthermore the number of triggers (e.g. buttons, gestures) a device supports, is a very important factor for the ease of use, because they are used for activation of the various functions of the VR system. We investigated several devices, currently available on the market, but finally had to develop our own device, the Flystick, because none of the commercially available devices fulfilled all requirements. The Flystick has a very handy shape, low weight, is robust and features three buttons, which are sufficient for most applications. The Flystick is equipped with a Polhemus tracking sensor (see color figure 1).

In the context of design review, there are several requirements concerning the output devices. Since multiple persons attend a meeting, single person devices like HMD, or Boom are not suitable. Furthermore it should be possible to display the models in a 1:1 scale and to interact with them in an intuitive way. Here, CAVE-like environments, for example Powerwall, CAVE and Virtual Tableare well suited.

Taking all advantages and disadvantages into account, Powerwall and the Flystick were chosen as hardware interfaces for the prototype implementation.

4 Basics

In general, each function carried out using a given user interface, can be seen as a sequence of atomic elements chosen from a predefined set of possible actions. Therefore we have to investigate how people act and afterwards which minimal set of atomic actions is required to fulfill all tasks.

To understand how people carry out actions, we take the action model of Norman [Nor90] into account. It consists of seven stages. His assumption that even complex interaction tasks can be broken down into basic tasks is consistent with the taxonomy of interaction tasks for computer graphics interfaces, suggested by Foley et. al. [FvDFH90]. Initially they were targeted to desktop applications, but several researchers have shown that this assumption holds for VR user interfaces too [Min97, PWBI97]. The following five basic interaction tasks are proposed:

1. **position:** The scope of the position-task is to move objects from one place to another
2. **selection:** The selection task is defined as choosing an element from a choice set. Foley distinguish between the selection of *variable sized set of choices* and the selection of *relative fixed-sized choice sets*. The first denotes object instances choice sets and the latter command and attribute choice sets, like menu selection

3. **quantify:** The quantify interaction task involves specifying a numeric value between a minimum and maximum value
4. **text:** The text input task entails entering character strings to which the application does not ascribe any special meaning. Thus, typing a command name is not a text-input task
5. **orient:** The orient task is defined as orienting objects in space

5 Realizing the BIT's

In the following chapters, we will discuss, how the BIT's should be realized, to fulfill the requirements of the described application domain and the proposed input and output devices. Nevertheless the presented concepts are general enough, to be adapted to other domains.

5.1 Positioning and Orienting

Especially in VR, there is no real separation between the position task and the orient task and most interaction techniques realize both of them. Therefore, both tasks will be described together in that section.

There are various metaphors known from literature [Bow97, Min97], for positioning and orienting the model. Nevertheless we introduced a technique called *World Point Grab*, which turned out to be very easy to use: The user grabs a point in the scene, which is not necessarily covered by an object. Then he moves his hand and the whole scene is moved accordingly. The grabbing and releasing is triggered by an event, like pressing a button on the input device. The center of rotation is the grabbed point. With this technique the user can interact in a very natural and direct way with the given model. One disadvantage is that during a single grab-move-release sequence, the maximum displacement is bound by the size of the user's arm. Larger displacements need numerous grab-move-release sequences, but for models of the size of a car body this is still feasible.

5.2 Object Selection

In our scenario object selection is currently needed for changing the visibility of the selected objects, as well as placing data probes in the scene to acquire the according simulation values.

In general we can distinguish between three different types of techniques. The most simple technique is "grab and select", where objects are selected by grabbing them. If objects are out of reach, arm extension, or GoGo techniques can be applied [Bow97].

Image plane techniques, like *Sticky finger*, or the *framing hands* [Pie97] are another possibility to select objects. Here the line of sight between the user's eye, the real hand or input device and the objects is used to determine the selection. For unambiguous selection results, an accurate tracking is needed and

the devices have to fulfil several requirements (e.g. a device with a small tip for Sticky finger).

The third well-known technique is ray casting, where a virtual laser beam is emitted from the hand in the direction the user points to. An object is selected, if the beam penetrates it and the user triggers an event, like pressing a button on his input device. Using this technique, it is very intuitive to select objects, which are not too far away. To improve accuracy when selecting distant objects [Fit54], the beam can be replaced by a cone (spotlight-technique).

Furthermore ray casting using a beam is the only technique, where it is possible to not only select objects, but also to select a single point on the surface of the models. In the context of simulation visualization, this allows to place a data probe on the surface and acquire the corresponding value. Because of the small size of the beam, the position of the probe is definite.

We choose the ray casting technique for our implementation, because it is a reliable technique, easy to use and it supports data probing. Consequently, only one metaphor for *relative fixed-sized choice sets* is needed.

5.3 Menu selection

Menu selection is the central part of any system with large set of functionality, because it is needed to switch between the various functions in a very simple and fast manner. Especially with command selection, speech input comes in mind. Speech recognition heavily matured during the last few years, but based on some experiments with commercially available software, it turned out, that these systems are still not robust enough for that type of application. Furthermore additional devices have to be carried around during the design review. For that reason we stick to a software only solution, using menu systems for the selection of commands and the invocation of various system functions.

Traditional versus pie menus Generally we can distinguish between two types of menus: 3D menus and 2D menus. 3D-menus are part of the scene, while 2D menus are rendered as overlay and therefore never encounter the problem of penetrated by the scene. The following concepts are based on 2D menus, but can be easily adapted to 3D menus.

Our first approach for the layout of the menu was to follow well-known concepts of desktop user interfaces. We introduced a menu system, which looked similar to the Windows(tm) menu system. The advantage is obvious: Since most people are used to Windows(tm) and it's applications, they will not have a long training period, to understand how they have to use that kind of menu system. It turned out, that the familiarity could not compensate the biggest disadvantage, its usability. Selecting a specific menu item required a very exact positioning of the virtual cursor and therefore performance and acceptance were very low. Nevertheless when used in a desktop style scenario with a mouse, that type of menu system was easy to use.

To avoid the need for high accuracy, we came up with pie menus, first introduced by [CHWS88] for desktop applications. A pie menu is defined as a circle

with 4, or 8 evenly distributed menu items (see color figure 5). When opening the menu, the cursor is in the center of the circle. The advantage of that type of menu is that all items are at the same distance from the initial position of the cursor and that the size of an entry increases with the distance from the center. Each menu item can be a button, or a link to a submenu. The type of the item is visually encoded in the menu.

Any simulation data set used in a Design Review, consists of several 100K polygons, which decreases rendering speed. Since the visual update of menu and cursor are bound to the framerate, interaction gets more difficult. Replacing the whole scene with a texture, or bitmap during menu selection, raises the frame rate up to 60 frames per second and interaction gets much easier.

The Speed-Bar In most software systems, only a few functions are always needed. In the simulation visualization scenario, data probing and playback control are often in use and activated in alternated order. Using the menu to switch between these modes is too costly and takes too much time. Therefore we need a possibility to rapidly switch between 2, or 3 functions, avoiding the standard menu selection. For this, we developed the so called *speed-bar*. The speed-bar stores the last recently used (LRU) functions and when activated, the user can choose one from this set. This is by far more flexible than a fixed set of functions, defined by the user.

The speed-bar is activated and displayed, as soon as the user points the flystick away from the projection screen and presses the menu button. The last activated function is selected. Through a horizontal translation, previous activated functions can be accessed. As soon as the button is released, the selected function is executed. Especially when only two functions are used in an alternating way (see above), activation of the previous function is straightforward: First point device away from projection screen, then click the button. Using the flystick as input device, the act of "pointing away" is very easy to carry out and supported by the proprioception.

Furthermore, the direction the user points to could be taken into account, to call other functions than the speed-bar.

5.4 Quantify

In that section we will present two interaction techniques for the quantification-task. Both are targeted to the interaction with simulation data, but can be used for other purposes too.

Color-Mapping A basic requirement is the ability to change the color mapping of the simulation values in real-time. For this we need a color gradient and several sliders to modify the mapping. Since the color gradient is one of the most important tools for evaluation of simulation data, it should be always visible.

The general idea to realize that requirement is using a geometrical objects, a pillar with connected handles. The height of the pillar represents the range of

possible values, where the top of the pillar is the maximum value and the bottom the minimum value. For CAVE style environments, the pillar is attached to the cart. The cart is part of the flying carpet paradigm, introduced by [Zac96]. The advantage is, that the pillar resides roughly at the same position, even when the user turns his head, or moves around in front of the projection screen. When using HMD, the pillar should be attached to the camera to stay in view. One way to move the handle could be by grabbing and dragging it. For this, the pillar has to be in reach of the user, which is not always given, especially not in front of a large projection screen.

Thus, we use a laser beam for selecting the handle. Nevertheless it is too complicated to switch the beam on via a speech command, or menu item, and too disturbing, when the beam is always visible. Therefore we tried to adapt the way people act in real world to the virtual world: If someone wants to "select" an item in the distance, he points at it. Taking this into account, the beam is automatically switched on and off, depending whether the user points to the pillar, or not. Technically, the intersection of the beam and the bounding box is tested. For an improved ease of use, we scaled the bounding box by a factor of 1.5. Since the pillar is located at the right side of the screen, it is out of reach of unintentional selection. Then as soon as the beam gets visible, only the manipulation of the pillar is possible and any other function of the system is disabled. Consequently, all buttons of the interaction device change their behavior. To be consistent, the button, which moves the scene, is used for dragging the pillar handles.

Since the pillar and its handle should not waste too much screen space, their maximum size is limited. On the other hand, small objects are hard to select and even when pressing the button on the device, the device slightly shakes and the beam moves to another place. Therefore we are not using the selection paradigm "click to select", known from desktop software applications, where the cursor has to point to an object, when the button is pressed. Instead we use "click and drag to select". Here, the user presses the button and then the first object is selected, which is penetrated by the beam. Then according to the beam, the handle is moved up and down. As soon as the user releases the button, the selection ends.

We implemented that color gradient as a colored pillar, with low values (blue) at the bottom and high values (red) at the top. For changing the color mapping, two handles are connected to the pillar, which represents the lower and upper boundary of the color gradient. All values smaller than the lower boundary, are rendered in blue, all values greater that the upper boundary, are rendered in red. This enables the user to squeeze the color gradient to improve investigation of smaller intervals. A third handle controls the transparency-boundary: All values below or over that boundary, are rendered transparent (see see color figure 3 and color figure 4).

Since the mapping should change in real-time, it is realized through texture coordinates referencing a 1D texture, the color gradient. As soon as the mapping changes, the OpenGL texture matrix is adapted accordingly. This is a very fast and cheap operation.

Simulation-control Since we have to visualize simulation results with various time steps, we need a possibility to switch from one time step to another and also to continuously play back the whole set of time steps at different speeds.

Again, we tried to adapt real world metaphors for the use in virtual environments: In the real world, a jog-dial can be found on a number of video recorders, to control the playback-speed of the video. The jog-dial is a rotational button, where the rotation is limited to typically +/- 90 degrees. The speed of the playback is mapped onto the angle and the direction on the sign of the angle.

The Jog-Dial for VR is a semicircle and an indicator, both attached to the virtual hand of the user (see color figure 2). The semicircle represents the range of possible values and the indicator points to the current value. The indicator rotates around the center of the semicircle and is controlled intuitively by rotating the wrist around the axis defined by the forearm. In contrast to the pillar, the Jog Dial is activated through a speech command, or menu item. When activated, a button on the device controls the jog-dial. As soon as the button is pressed, the indicator rotates according to the wrist, until the button is released. Furthermore we can take advantage of proprioception, when resetting the indicator to a default position (e.g. center), each time the button is pressed. Then the user always knows the start-value of the indicator, and can easily control the jog-dial, even when not in sight. Bringing the jog-dial back in sight means levering the forearm, which may result in earlier fatigue.

For the playback control, we did not use a linear mapping of angle to speed. Instead the semicircle was divided into seven evenly sized areas. If the pointer is in the middle area (green), the animation stops. Through the behavior mentioned above, a short press-release cycle of the button is enough to set the indicator to the default position and to stop the animation. The following areas (yellow) return a fixed value and the outmost areas interpolate linearly between two given values. In our case, the yellow area sets the speed to one time-step per second, which can be either used for continuous playing, or for going through the data set step by step. For the continuous-mode, the indicator is set to the yellow area and the button is released. For step-mode the indicator is set to the yellow area and as soon as the next time step is displayed, the indicator is set to the center area and the button released. The outmost area returns speed values between 1 and 10 frames per second for rapid playback.

5.5 Text

In the context of design review applications, the text task is used for the documentation and therefore we need the possibility to insert a large amount of text in a very natural way. Since we abstain from speech input, the efficient and easy way for text input is using a keyboard and window on a separate computer. Since many text lines are typed, it does not make real sense to display the text directly in the scene.

6 Results

In this chapter we will shortly describe the developed prototype, based on the results discussed so far. The target platform was a SGI ONYX2 with Infinite-Reality Graphics, a large screen projection using active stereo (approx. 3m by 2m), a magnetic tracking system and the Flystick as an input device. For text input a PC was available.

Beside the user interface, the integration of the simulation results in the system was an important aspect. This was done through the finite element kernel, developed by Fraunhofer-IGD. FEK is based on the Model View Controller (MVC) concept and is able to store various kinds of time dependant data, ranging from 2D elements in 3D space (surface models), to 3D elements in 3D space (volume models) with different types of attributes attached to each node and/or element. FEK features several visualization objects (VO), which transform the raw data into visible objects. The result of that transformation depends on the type of VO. There are VO's for surface models, which display 1D values as colors, or textures, 3D values as arrows, etc. VO's for flow simulation, cutting planes and iso-surfaces are currently under development. Furthermore FEK is able to split a data set into several groups, where one group represents a specific part of the simulation data. For example, when a simulation is based on a whole car body, the various parts (e.g. door, hood) can be assigned to different groups. Then any manipulation can be done locally on these groups, like changing visibility. Otherwise any manipulation would affect the whole car body.

Since we have numerous possible functions, but only a limited set of buttons on the Flystick, the buttons can trigger one out of n possible functions. Which actual function is chosen is determined by the function-mode. Using the menu system, the user can activate different function modes, like data probe. The active mode is shown on the left side of the screen enriched with information supplied by these functions. In the case of data probe, the value is displayed. The Flystick offers three buttons. Two buttons are predefined and never change their behavior. The third one, the "action button", is used to activate the different functions according to the current function mode, like setting a data probe, or manipulating the jog-dial. The first button is used to move the scene using the *grab world point* metaphor. The second button opens the pie-menu as soon as it is pressed and closes and executes the selected item as soon as it is released.

Data probes can be placed using ray casting and an additional curve is displayed, which represents the values of that probe as a function of time (see color figure 6). Playback and changing the mapping are implemented according to the concepts presented in the previous chapters. For documentation purposes, the ability to place markers using ray casting and to generate snapshots was added to the system.

7 Conclusion

In this paper we presented a concept for an intuitive VR user interface for interaction with simulation data. We compared different approaches, to realize the

basic interaction tasks (BIT). We developed new metaphors for the quantification, the jog-dial and the pillar-slider. Known menu concepts were evaluated and adapted to VR. Finally the whole functionality was realized in a prototype implementation.

For the future it could be very interesting to add more functionalities to the system and to adapt the user interface to the increasing requirements.

8 Acknowledgments

Parts of this project are funded by the European Commission (AIT DMU-VI, Brite Euram Project BRPR-CT97-0449). We thank all our colleagues and students at our laboratory, especially H. Haase, M.Lux, D. Reiners and G. Vo, without their work we would not have been able to achieve the results presented herein. Finally we thank D. Keller for his valuable input and feedback.

References

[Bow97] D.A. Bowman. An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. In *Symposium on Interactive 3D Graphics*, 1997.

[CHWS88] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An empirical comparison of pie vs. linear menus. In *Proceedings of SIGCHI 1988*, pages 95–100, 1988.

[Fit54] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, (47):381–391, 1954.

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics, Principles and Practice.* Addison-Wesley Publishing Co. Inc., 2 edition, 1990.

[Min97] M. Mine. Moving objects in space: Exploiting proprioception in virtual environment interaction. In *Proceedings of SIGGRAPH 1997*, 1997.

[Nor90] D. Norman. The design of everyday things. *Currency Doubleday*, 1990.

[Pie97] J. Pierce. Image Plane Interaction Techniques in a 3D immersive environment. In *Symposium on Interactive 3D Graphics*, 1997.

[PWBI97] I. Poupyrev, S. Weghorst, M. Billighurst, and T. Ichikawa. A framework and Testbed for stying manipulation techniques for immersive VR. In *Proceedings of ACM Conference VRST*, pages 21–28, 1997.

[Zac96] G. Zachmann. A Language for Describing Behaviour of and Interaction with Virtual Worlds. In *Proceedings of ACM Conference VRST*, July 1996.