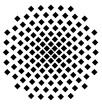

Universität Stuttgart

The Design and Implementation of a Presentation System for Interactive 3D Graphics Applications

S. Stegmaier T. Klein M. Strengert T. Ertl

Bericht Nr. 2005/05
Juli 2005

Institut für Visualisierung und Interaktive Systeme
Fakultät für Informatik, Elektrotechnik und Informationstechnik
Universitätsstraße 38, 70569 Stuttgart

The Design and Implementation of a Presentation System for Interactive 3D Graphics Applications

S. Stegmaier T. Klein M. Strengert T. Ertl
Institute for Visualization and Interactive Systems
Universität Stuttgart, Universitätsstraße 38
70569 Stuttgart, Germany

Abstract

We present the design and implementation of a stand-alone system for presenting interactive 3D graphics applications and arbitrary multimedia contents to a public audience. The description includes technical details regarding the construction of a sturdy case to accommodate touchscreen, PC, video projector, and sound equipment, and details involved in the design of the presentation software. The presented system was evaluated during a week-long exhibition with several hundreds of users.

1 Introduction

The University of Stuttgart is a state university mainly funded by the public. To express its appreciation for this funding, it was decided to organize a series of talks and exhibitions with hands-on exhibits on a popular science level—a "Science Year"—to inform the public about research projects being financed by the tax payers' money. The main event was the so-called "Summer of Science" which included week-long exhibitions of various university institutes, organized centrally located collection of booths with posters explaining the respective institute's research goals. The Institute for Visualization and Interactive Systems (VIS), the university's computer graphics group, decided to present a selection of real-world applications developed at the institute in recent years and found in the car manufacturing industry, medicine, and other areas. Although the demonstrations should be available continously, presence of researchers had to be reduced to half a day each day. This lead to various problems: How can interactive 3D graphics applications be presented to an audience without knowledgeable staff being around? How can users be prevented from closing the application and gaining access to the machine, maybe shutting it down? How can keyboards, displays, and computers be secured against vandalism? What was needed was a cabinet similar to those found in stations and airports for looking up timetables or making reservations—locked boxes with sturdy touchscreens. However, building a system capable of presenting arbitrary applications with complex user interfaces poses several interesting hardware and software problems which we will address in this paper, based on our experiences with building and operating such a system during the "Summer of Science 2004".

2 Related Work

Several commercial and open-source software solutions are available for creating and presenting slide presentations. Most well-known are Microsoft's *PowerPoint* [1] and Sun's *OpenOffice* suite [2]. Both solutions enable the user to create presentations suitable for a public audience. The systems allow for the inclusion of audio and video files and even allow for the execution of arbitrary external applications. These systems may therefore appear as suitable choices for the intended scenario. However, PowerPoint is not available on Linux, our target platform, and both systems cannot present external applications without overwhelming inexperienced users by their complex user interfaces nor keep them from tempering with the system in an undesired way. Both systems can, therefore, at best give hints at how a presentation system might be organized that meets our requirements.

3 Requirements

Both hardware and software requirements were defined for the desired presentation system. The case had to accommodate both an 18 inch touchscreen (which was already available), a PC tower case, a moderately-sized video projector and sound equipment (for video playback and acoustical feedback). Whenever possible, the devices should be hidden from the user to prevent vandalism. There was neither

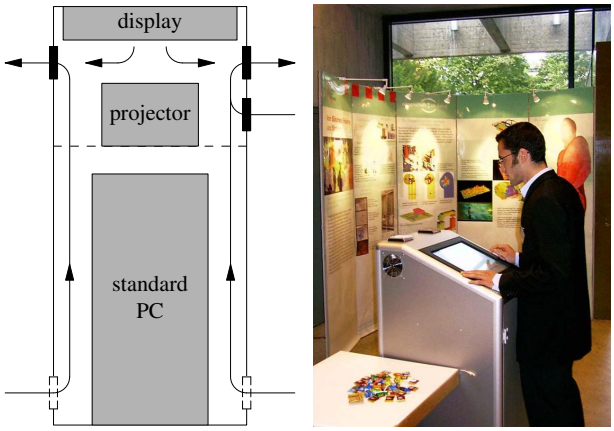

Figure 1: The hardware component of the proposed presentation system. Left: Schematic drawing of the cooling; right: Photograph of the actual box operated during the exhibition (courtesy IFF, University of Stuttgart).

a need for a keyboard nor a mouse since the system should only allow for remote administration via a network connection. The projector was included to attract people and to make it easier to gather a view of the running application once a crowd had formed around the display. The software requirements were dictated by the applications to be presented and the intended audience: Linux operating system, able to present arbitrary applications with (partially) hidden user interfaces, e.g. menu bars, but without requiring any modifications. The latter issue is of special importance since the system will only be accepted if existing applications can be integrated as-is. Finally, the system was to be operated by researchers of various institutes which added the requirement of being able to start up upon powering it on and shutting it down from a key-protected slide from within the running presentation without requiring user logins.

4 System Design

4.1 Hardware Design

The basic case was build to specifications by a professional carpenter. It included an aluminium frame with the side walls made from sturdy 4 mm plastic enclosed in 1 mm thin sheets of aluminium. The top and the lockable front were built from covered particle board. For cost reasons, the final modifications to the case were made by university technicians and included all aspects of integrating the electronic devices to be operated during the presentation. Since both the touchscreen and the projector generate a considerable amount of heat, providing sufficient cooling poses the main difficulty in designing the interior. In our solution (Fig. 1, left) cool air flows through large inlets near the bottom and is then being transported upwards through a perforated metal plate accommodating the projector into a low-pressure compartment generated by two 220 V/18 W fans. Another fan induces vortices between the projector and the case back wall so that projector models with cooling fans installed next to the projector lens can be used without overheating. Fig. 1, right, shows the actual box as it was operated during the exhibitions.

4.2 Software Design

As was described in Sec. 3, the desired presentation software has a simple requirement: It must be able to run arbitrary applications without any modifications and without overwhelming the user with complex user interfaces. We achieve this property by defining the screen area covered by a presentation slide as a set of windows—a *window mask*—covering the whole desktop and always being on top of the stacking order. The center part of this window mask (in the following called the “demo window”) can be unmapped to open the view to selected parts of underlying windows of arbitrary external applications (Fig. 2). For 3D graphics applications, the underlying external application is typically composed of a large OpenGL drawable and surrounding user interface elements. Thus, these applications are positioned so that the user interface is clipped either at the desktop borders or the window mask. Fig. 3 illustrates this process by successively increasing the transparency of the window mask, thereby exposing the application area to be seen by the user. Of course, upon unmapping the demo window any events occurring in the exposed area will be passed to the underlying window which enables the user to interact with external applications without losing the impression of viewing a presentation. For this to work, the window mask color must match the ex-

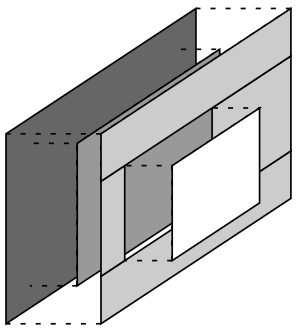

Figure 2: The basic idea of the presentation system software: A window mask exposes only the part of the application’s user interface that is intended for the audience. The demo window is shown in white.

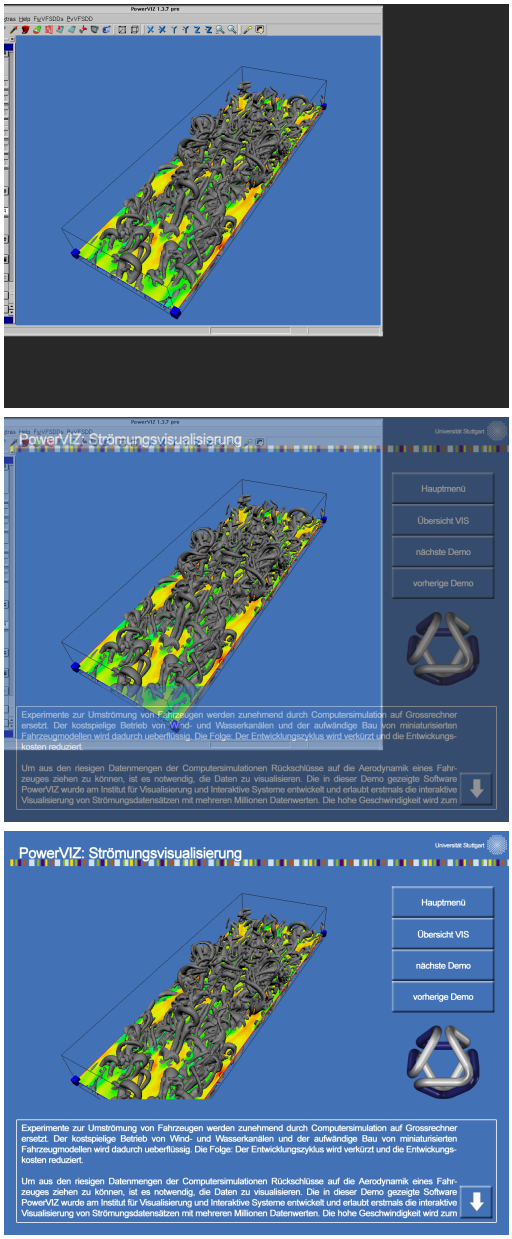

Figure 3: Clipping the user interface. Top: application after being started with the preload library (background color, position, and size are being set, UI partially clipped at desktop); bottom: final result as seen by the user (UI clipped at window mask).

posed window's background color. If the application window cannot be scaled to fill the demo window, the root window background color additionally must match the window mask color. Since in general these colors will not match, we provide a non-intrusive solution taking advantage of the operating system's dynamic loading and linking functionality [4] to override the background-setting functions and to simultaneously disable window manager control by setting the `override_redirect` flags during window creation so that the top-level window's size and position can be scripted [5]. We found this solution to be reliable even for complex commercial applications.

4.2.1 Slide Creation

The concept of a WYSIWYG interface for slide creation was discarded since this functionality coins the appearance of slides, leading to presentations often lacking a unique layout and design. Instead, arbitrary image manipulation programs are used to create the slide images and active areas (buttons) are specified in a separate text file written in a simple slide description language. The slide image is automatically cut into subimages suitable for creating the window mask with a cropping script generated by the presentation software. If required, smooth transitions between the *application-loading* state and the *application-running* state can be obtained with an integrated screenshot function that captures the demo window area right after the demo window has been unmapped and the application becomes visible. Fig. 4 shows an example slide for a flow visualization demonstration. The captured demo window has been extended by a pointing-hand image and an explanatory text to better indicate the loading of an interactive demo.

4.2.2 Slide Description

The appendix gives the complete slide description used during the "Summer of Science 2004" presentation, starting with area definitions for the various buttons and then proceeding by specifying the individual slides. Area definitions are given in the usual X Window geometry specification format of the form $W \times H + X + Y$ where W and H specify the area width and height, respectively, and X and Y denote the position. Thus, e.g., `Down = 83x83+1155+899` gives the area specification of the arrow-labeled button in the lower right corner of Fig. 4. The area definitions are followed by an obligatory slide description for each slide which assigns a unique slide number and the actions to be performed when certain areas are clicked[1]. The definition for the slide sequence corresponding to the application shown in Fig. 4 is given in the section labeled `POWERVIZ`. We will examine the file structure at this example. The sequence comprises slides #4 and #5. These slide identifiers are assigned explicitly by giving the respective identifier right at the beginning of the corresponding slide description. The text following the identifier and up

[1] Our touchscreen is limited to moving the mouse pointer and to simulate clicks and double-clicks; mouse events are, therefore, the only events that have to be processed.

Figure 4: Example slide used during the exhibition. Active areas are depicted as 3D buttons, the text box at the bottom displays a scrollable text explaining the current application. The center text reads *Please wait ... interactive demo is being loaded*.

to the closing bracket specifies the image data to be used for composing the slide. In our example, this text reads `slides/slidePowerViz_0_` which indicates that the slide is made up of all the image files in the directory `slides` beginning with the prefix `slidePowerViz_0_`.

The remaining lines of the slide description specify actions. These actions are enclosed by angle brackets < and > and can be stand-alone to indicate *autostart actions* run automatically when the slide is loaded or *user actions* (indicated by curly braces { and }) to be executed upon user interactions. In the example, the user action `{Down $5 <0,./click,-1>}` asks the system to execute the program `click` in the current directory 0 s after the area designated by the variable `Down` has been clicked by the user and then to jump to slide 5. The number -1 indicates that the program must not be kept running when switching to another slide or when hitting the `Down` area again. In contrast, the autostart action is supposed to expose the application `powerviz.1.4.6.linux` found in the working directory after a loading time of 7 s and to keep the application running when switching to the next slide (since this slide contains the second half of the explanatory text shown in the text box at the lower border). The remaining slides are specified analogously. This also applies to the service slide (section `CODE LOCK`), a slide reached by clicking a tool symbol on the presentation main page and intended to provide the booth personnel with some means for rebooting the hosting machine or for shutting it down. However, obviously this functionality must be inaccessible for the audience to prevent misuse. We will shortly demonstrate a way to implement this functionality without having to resort to specifically tailored stand-alone applications. In contrast, our solution is based exclusively on small and easy to maintain shell scripts.

4.2.3 Implementing Special Functionality

The basis for the service slide again is laid by providing image data indicating active areas, in this case an image of a code lock (Fig. 5), and by defining the respective regions for the buttons. The latter is accomplished by the definitions of `Key0` to `Key9` in the provided description file and the definitions for the asterisk and hash symbols. The idea now is to define a number of codes (number keys to be pressed in sequence) each provoking a certain functionality. Entering a code is terminated by pressing the hash key, the asterisk is assigned the functionality to start over in case erroneous data was entered. Independently of whether a number key or some other key was pressed, the same shell script shown in Fig. 6 is executed and the individual invocations only differ in the arguments passed. For number keys the corresponding numerical value is passed. Examining the shell script reveals that in this case the entered code is successively compiled in a code file by appending the entered digit. When clicking the asterisk key, this code file is simply cleared—the input process starts anew. When the user indicates the input code is complete by pressing the hash key, the code is read from the code file and compared to a list of

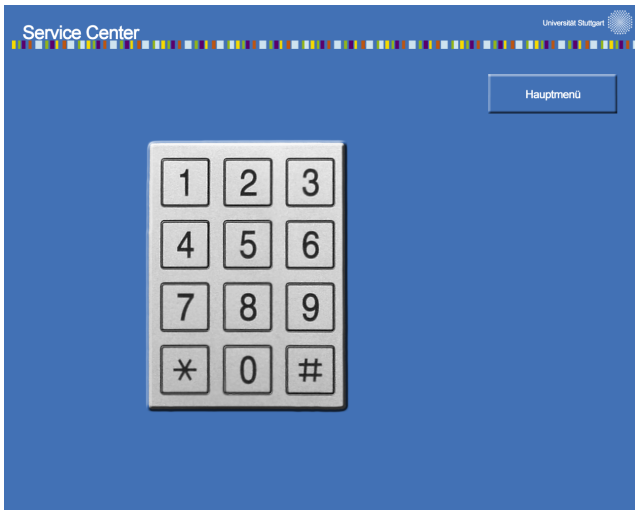

Figure 5: Service slide. Entering certain codes provokes certain actions like rebooting or halting the machine hosting the presentation system.

```
#!/bin/sh

CODEFILE=secret.code

SHUTDOWN=47114711
REBOOT=08150815

./click

case "$1" in
    [0-9]   )
        echo -n $1 >> $CODEFILE
        ;;
    a       )
        echo -n > $CODEFILE;;
    b       )
        CODE=‘cat secret.code‘
        echo -n > $CODEFILE
        case $CODE in
        $SHUTDOWN   )
            killall X && \
            /sbin/shutdown -h now
            ;;
        $REBOOT     )
            killall X && \
            /sbin/shutdown -r now
            ;;
        *           )
            true
            ;;
        esac;;
    *       )
        true;;
esac
```

Figure 6: Shell script implementing service functionality.

predefined number sequences. If a match is found, the corresponding action is performed. The implementation shows that by taking full advantage of the Unix programming environment the chosen design allows for integrating special functionality quickly and without burdening the presentation developer with the need for programming skills beyond standard knowledge.

4.2.4 Managing External Applications

Managing external applications is the most delicate part in a system that is being operated without staff and potentially malicious users trying to bring the system down to its knees by, e.g., starting and stopping applications in rapid order. This is caused by the fact that unreliable process termination will in the long term result in scarce resources and, therefore, render the system into an unusable or unstable state. We took great efforts to prevent these problems. In our solution, the presentation system spawns a new child C_0 for each external application to be executed using the `fork(2)` system call. The child C_0 then becomes the session leader by calling `setsid(2)` to initiate a new process group. It then spawns another child C_1 whose sole purpose is to run the external application via `exec(2)` and terminates afterwards. After spawning C_1, child C_0 suspends its execution for the number of seconds specified in the slide description. Setting this time span is the user's task and depends on how long the external application takes to be up and running. C_0 then opens a new display connection and, if required, unmaps the demo window. Finally, C_0 calls `pause(2)` to suspend its execution until a parent signal is being received and then terminates. Running external application this way is fairly complex. However, the great advantage of this solution is that now the operating system keeps track of all subprocesses spawned be the external application. A single `killpg(3)` is then sufficient to reliably clean all processes related to the external application (see [3]).

5 Evaluation

The presented presentation system was evaluated during a week-long exhibition. Three university institutes contributed an overall of 7 videos and 5 interactive 3D graphics applications, including volume rendering of complete human datasets, live visualizations of flows around car bodies, visualizations of cosmic events, and architectural visualization. Due to the limited interaction possibilities of the touchscreen device, in most cases user interactions were limited to rotations and zooming. During the exhibition, all user interactions were logged in order to obtain meaningful usage statistics. Overall more than 2,500 application invocations were processed. Almost 700 audio-visual presentations explaining research fields were watched by the audience and the most popular application (an example of flow visualization) was started several hundred times. The system seemed to be especially attracting to children and school classes which seemed to be able to navigate through the slide hierarchy intuitively without prior explanations. In contrast, older generations tended to be reluctant to use the system, astonishingly often fearing to break the device. During the whole week, there was never any malfunction or a system crash. The casing proved to be too heavy for being moved accidentally. The fans—while being quite noisy in an office atmosphere—did not bother in an actual presentation environment which is very noisy anyway. The cooling proved sufficient and did not cause any overheating. Due

to the robustness of the casing and touchscreen, no damage was made at all although the system was located mostly unattended in a public place.

6 Conclusion and Future Work

We have given an overview of the design of a standalone presentation system for arbitrary multimedia content, including interactive OpenGL-based applications for 3D scientific visualization. The system—both hardware and software—was build from scratch. The system includes a 4,000 Euro touchscreen mounted to a 1,000 Euro casing (only considering the carpenter's wages) and is, thus, not cheap. However, the resulting system proved very flexible and professional in a real-world exhibition and is well-worth the money. As a conclusion, we can thoroughly recommend the given procedure to anyone with an interest in the public presentation of interactive graphics applications, be it for product marketing or educational purposes.

Creating uniform layouts for very large presentations using general image manipulation software may be quite cumbersome. However, first results show that the slide creation can be completely scripted based on LaTeX templates, which has the additional benefit of allowing the easy integration of complex mathematical expressions into the explanatory texts given on the slides.

Acknowledgments

We are grateful to Peter Burger for coordinating the case construction and to Erwin Beck for integrating the electronics into the case. The touchscreen was gratefully provided by Hanns Ruder, University of Tübingen.

References

[1] Microsoft Corporation, PowerPoint Product Information, *http://www.microsoft.com/office/powerpoint/prodinfo*, 2004.

[2] Sun Microsystems, Inc., OpenOffice Impress *http://www.openoffice.org/product/impress.html*, 2004.

[3] W. R. Stevens, *Advanced Programming in the UNIX Environment,* Addison-Wesley, 1993.

[4] W. W. Ho and R. A. Olsson, "An Approach to Genuine Dynamic Linking," *Software, Practice and Experience,* Vol. 21 (4), pp. 375–390, John Wiley & Sons, 1991.

[5] A. Nye, *Volume 0: X Protocol Reference Manual,* X Window System Series, O'Reilly & Associates, 4th Edition, 1995.

Appendix — Example Slide Description File

```
#
# NOTE: DO NOT RUN ANY ACTIONS IN THE
#       BACKGROUND!THE PRESENTATION
#       TOOL TAKES CARE OF THIS!
#

# Slide format:
# <slide number> <file mask>
Service = 80x110+1200+915
Up = 83x83+1155+742
Down = 83x83+1155+899
MainMenu = 260x80+983+140
AboutVIS = 260x80+983+220
NextDemo = 260x80+983+300
PrevDemo = 260x80+983+380
MainDemo1 = 548x325+36+140
MainDemo2 = 548x325+404+309
MainDemo3 = 548x325+36+478
MainDemo4 = 548x325+404+647
MainDemo5 = 548x325+36+222
MainDemo6 = 548x325+404+301
MainDemo7 = 548x325+36+560
MainVis = 376x320+20+463
MainSGS = 376x320+452+532
MainITE = 376x320+887+463
VolumeDecrease = 80x80+983+620
VolumeIncrease = 80x80+1163+620
Key0 = 100x100+445+690
Key1 = 100x100+318+310
Key2 = 100x100+445+310
Key3 = 100x100+571+310
Key4 = 100x100+318+436
Key5 = 100x100+445+436
Key6 = 100x100+571+436
Key7 = 100x100+318+563
Key8 = 100x100+445+563
Key9 = 100x100+571+563
Asterisk = 100x100+318+690
Hash = 100x100+571+690

# ###################################
# VIS MAIN PAGE
# ###################################

[0 slides/slideMain_0_]
#{MainMenu $21 <0,./click,-1>}
{Service $24 <0,./key.sh a,-1>}
{MainVis $1 <0,./click,-1>}
{MainSGS $2 <0,./click,-1>}
{MainITE $3 <0,./click,-1>}

# ###################################
# VIS MAIN PAGE
# ###################################
```

```
[1 slides/slideVisMain_0_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $2 <0,./click,-1>}
{NextDemo $3 <0,./click,-1>}
{MainDemo1 $4 <0,./click,-1>}
{MainDemo2 $6 <0,./click,-1>}
{MainDemo3 $9 <0,./click,-1>}
{MainDemo4 $11 <0,./click,-1>}
# ###################################
# SGS MAIN PAGE
# ###################################
[2 slides/slideSGSMain_0_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $3 <0,./click,-1>}
{NextDemo $1 <0,./click,-1>}
{MainDemo1 $16 <0,./click,-1>}
{MainDemo2 $18 <0,./click,-1>}
{MainDemo3 $14 <0,./click,-1>}
{MainDemo4 $20 <0,./click,-1>}

# ###################################
# ITE MAIN PAGE
# ###################################
[3 slides/slideITEMain_0_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $2 <0,./click,-1>}
{MainDemo5 $21 <0,./click,-1>}
{MainDemo6 $22 <0,./click,-1>}
{MainDemo7 $23 <0,./click,-1>}

# -----------------------------------
# POWERVIZ
# -----------------------------------
[4 slides/slidePowerViz_0_]
<7,./powerviz.1.4.6.linux,4-5>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $6 <0,./click,-1>}
{PrevDemo $11 <0,./click,-1>}
{Down $5 <0,./click,-1>}
# -----------------------------------
[5 slides/slidePowerViz_1_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $6 <0,./click,-1>}
{PrevDemo $11 <0,./click,-1>}
{Up $4 <0,./click,-1>}

# -----------------------------------
# STUDENTISCHE ARBEITEN
# -----------------------------------
[6 slides/slideStudent_0_]
<1,./mplayer_Stud,6-8>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $9 <0,./click,-1>}
{PrevDemo $4 <0,./click,-1>}
{Down $7 <0,./click,-1>}
{VolumeDecrease <0,./volume_decrease,-1>}
{VolumeIncrease <0,./volume_increase,-1>}
# -----------------------------------
[7 slides/slideStudent_1_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $9 <0,./click,-1>}
{PrevDemo $4 <0,./click,-1>}
{Up $6 <0,./click,-1>}
{Down $8 <0,./click,-1>}
{VolumeDecrease <0,./volume_decrease,-1>}
{VolumeIncrease <0,./volume_increase,-1>}
# -----------------------------------
[8 slides/slideStudent_2_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $9 <0,./click,-1>}
{PrevDemo $4 <0,./click,-1>}
{Up $7 <0,./click,-1>}
{VolumeDecrease <0,./volume_decrease,-1>}
{VolumeIncrease <0,./volume_increase,-1>}

# -----------------------------------
# VISIBLE HUMAN
# -----------------------------------
[9 slides/slideVisHum_0_]
<3,./vProgram,9-10>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $11 <0,./click,-1>}
{PrevDemo $6 <0,./click,-1>}
{Down $10 <0,./click,-1>}
# -----------------------------------
[10 slides/slideVisHum_1_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $11 <0,./click,-1>}
{PrevDemo $6 <0,./click,-1>}
{Up $9 <0,./click,-1>}

# -----------------------------------
# POINTCOUDS
# -----------------------------------
[11 slides/slidePoint_0_]
<12,./pointcloud,11-13>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $4 <0,./click,-1>}
{PrevDemo $9 <0,./click,-1>}
{Down $12 <0,./click,-1>}
#-----------------------------------
[12 slides/slidePoint_1_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $4 <0,./click,-1>}
{PrevDemo $9 <0,./click,-1>}
{Down $13 <0,./click,-1>}
```

```
{Up $11 <0,./click,-1>}
#-------------------------------------
[13 slides/slidePoint_2_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $1 <0,./click,-1>}
{NextDemo $4 <0,./click,-1>}
{PrevDemo $9 <0,./click,-1>}
{Up $12 <0,./click,-1>}


# ###################################
# SGS MATERIAL
# ###################################

# -------------------------------------
# OCTREE
# -------------------------------------
[14 slides/slideOctree_0_]
<1,./spp1103,14-15>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $2 <0,./click,-1>}
{NextDemo $16 <0,./click,-1>}
{PrevDemo $20 <0,./click,-1>}
{Down $15 <0,./click,-1>}
# -------------------------------------
[15 slides/slideOctree_1_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $2 <0,./click,-1>}
{NextDemo $16 <0,./click,-1>}
{PrevDemo $20 <0,./click,-1>}
{Up $14 <0,./click,-1>}

# -------------------------------------
# HIERACHICAL WORLD
# -------------------------------------
[16 slides/slideHWelt_0_]
<1,./mplayer_HWelt,16-17>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $2 <0,./click,-1>}
{NextDemo $18 <0,./click,-1>}
{PrevDemo $14 <0,./click,-1>}
{Down $17 <0,./click,-1>}
{VolumeDecrease <0,./volume_decrease,-1>}
{VolumeIncrease <0,./volume_increase,-1>}
# -------------------------------------
[17 slides/slideHWelt_1_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $2 <0,./click,-1>}
{NextDemo $18 <0,./click,-1>}
{PrevDemo $14 <0,./click,-1>}
{Up $16 <0,./click,-1>}
{VolumeDecrease <0,./volume_decrease,-1>}
{VolumeIncrease <0,./volume_increase,-1>}

# -------------------------------------
# FLOW AND MORE
# -------------------------------------
[18 slides/slideFlowCo_0_]
<1,./mplayer_FlowCo,18-19>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $2 <0,./click,-1>}
{NextDemo $20 <0,./click,-1>}
{PrevDemo $16 <0,./click,-1>}
{Down $19 <0,./click,-1>}
{VolumeDecrease <0,./volume_decrease,-1>}
{VolumeIncrease <0,./volume_increase,-1>}
# -------------------------------------
[19 slides/slideFlowCo_1_]
{MainMenu $0 <0,./click,-1>}
{AboutVIS $2 <0,./click,-1>}
{NextDemo $20 <0,./click,-1>}
{PrevDemo $16 <0,./click,-1>}
{Up $18 <0,./click,-1>}
{VolumeDecrease <0,./volume_decrease,-1>}
{VolumeIncrease <0,./volume_increase,-1>}

# -------------------------------------
# Mozart
# -------------------------------------
[20 slides/slideMozart_0_]
<1,./mplayer_Mozart,20>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $2 <0,./click,-1>}
{NextDemo $14 <0,./click,-1>}
{PrevDemo $18 <0,./click,-1>}
#{VolumeDecrease <0,./volume_decrease,-1>}
#{VolumeIncrease <0,./volume_increase,-1>}


# ###################################
# ITE MATERIAL
# ###################################

# -------------------------------------
# WAVES
# -------------------------------------
[21 slides/slideWellen_0_]
<1,./mplayer_Wellen,21>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $3 <0,./click,-1>}
{NextDemo $22 <0,./click,-1>}
{PrevDemo $23 <0,./click,-1>}

# -------------------------------------
# TOMOGRAPHY
# -------------------------------------
[22 slides/slideTomo_0_]
<1,./mplayer_Tomo,22>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $3 <0,./click,-1>}
{NextDemo $23 <0,./click,-1>}
{PrevDemo $21 <0,./click,-1>}

# -------------------------------------
# FIELDS
# -------------------------------------
```

```
[23 slides/slideFelder_0_]
<1,./mplayer_Felder,23>
{MainMenu $0 <0,./click,-1>}
{AboutVIS $3 <0,./click,-1>}
{NextDemo $21 <0,./click,-1>}
{PrevDemo $22 <0,./click,-1>}

# ###################################
# CODE LOCK
# ###################################

# -----------------------------------
# Code Lock
# -----------------------------------
[24 slides/slideCodeLock_0_]
{MainMenu $0 <0,./click,-1>}
{Key0 <0,./key.sh 0,-1>}
{Key1 <0,./key.sh 1,-1>}
{Key2 <0,./key.sh 2,-1>}
{Key3 <0,./key.sh 3,-1>}
{Key4 <0,./key.sh 4,-1>}
{Key5 <0,./key.sh 5,-1>}
{Key6 <0,./key.sh 6,-1>}
{Key7 <0,./key.sh 7,-1>}
{Key8 <0,./key.sh 8,-1>}
{Key9 <0,./key.sh 9,-1>}
{Asterisk <0,./key.sh a,-1>}
{Hash <0,./key.sh b,24>}
```