Paths of Virtu

A guide to the SGI© Virtu VN200 visualisation cluster at the Virtual Reality Center of Johannes Kepler University, Linz.

JOHANNES KEPLER
UNIVERSITY LINZ | JKU

Author: Johannes Zarl
This document was created from revision 332:333 on May 9, 2011.

Contents

List of Tables

List of Figures

1 About this document

1.1 Getting this document

The latest version of this document can be found at the following URL:
`http://vrc.zid.jku.at/studenten/dokumente/pathsofvirtu.pdf`

1.2 Reader Comments

Please send comments, corrections, and suggestions to johannes.zarl at jku.at.

2 System Information

2.1 Installed Libraries

Tables 1 and 2 give an overview of installed library versions and their locations. However, the `module` command (see section 3.1) will always give you just-as or more accurate information and is able to automatically adapt your environment for the libraries that you want to use. You can always get a list of installed libraries on virtu by issuing the following command:

```
user@virtu:~> module avail -l 2>&1 |sed -n '/\/jkuvrc\/modules:/,/^$/ p'
```

To get more information about a certain library package, use:

```
user@virtu:~> module show PACKAGE
user@virtu:~> module help PACKAGE
```

Library	Version	ICatcher	Virtu
bergen	0.4.1	X	X
CAVELib	3.2		X
ODE	0.8	X	X
	0.11.1		X
OpenAL	0.0.8	X	
	1.8.466		X
FreeALUT	1.0.1		X
OpenProducer	0.8.3	X	
OpenSG	1.4	X	
	1.6	X	
	1.8		X
	2.0.0		X
OpenSceneGraph	1.0	X	
OpenThreads		X	
trackdAPI	2.2		X
VRPN	07.26		X
VSS	4.0		X

Table 1: Installed libraries

Library	Version	Path
bergen		/jkuvrc/packages/bergen
CAVELib		/jkuvrc/packages/CAVELib/CAVE
ODE	0.7	/jkuvrc/packages/ODE/ode-0.7
	0.8	/jkuvrc/packages/ODE/ode-0.8
OpenAL		/jkuvrc/packages/OpenAL
FreeALUT		/jkuvrc/packages/OpenAL
OpenProducer		/jkuvrc/packages/Producer
OpenSG	1.6	/jkuvrc/packages/OpenSG/OpenSG-1.6
	1.8	/jkuvrc/packages/OpenSG/OpenSG-1.8
OpenSceneGraph		/jkuvrc/packages/OpenSceneGraph
OpenThreads		/jkuvrc/packages/OpenThreads
trackdAPI		/jkuvrc/packages/trackdAPI
VSS		/jkuvrc/packages/VSS

Table 2: Paths of libraries

2.2 Virtu Hardware

Virtu is a cluster containing a head-node (virtu itself) and 8 graphics-nodes, named n001 through n008. Each graphics-node runs an XServer which can be used for visualisation.

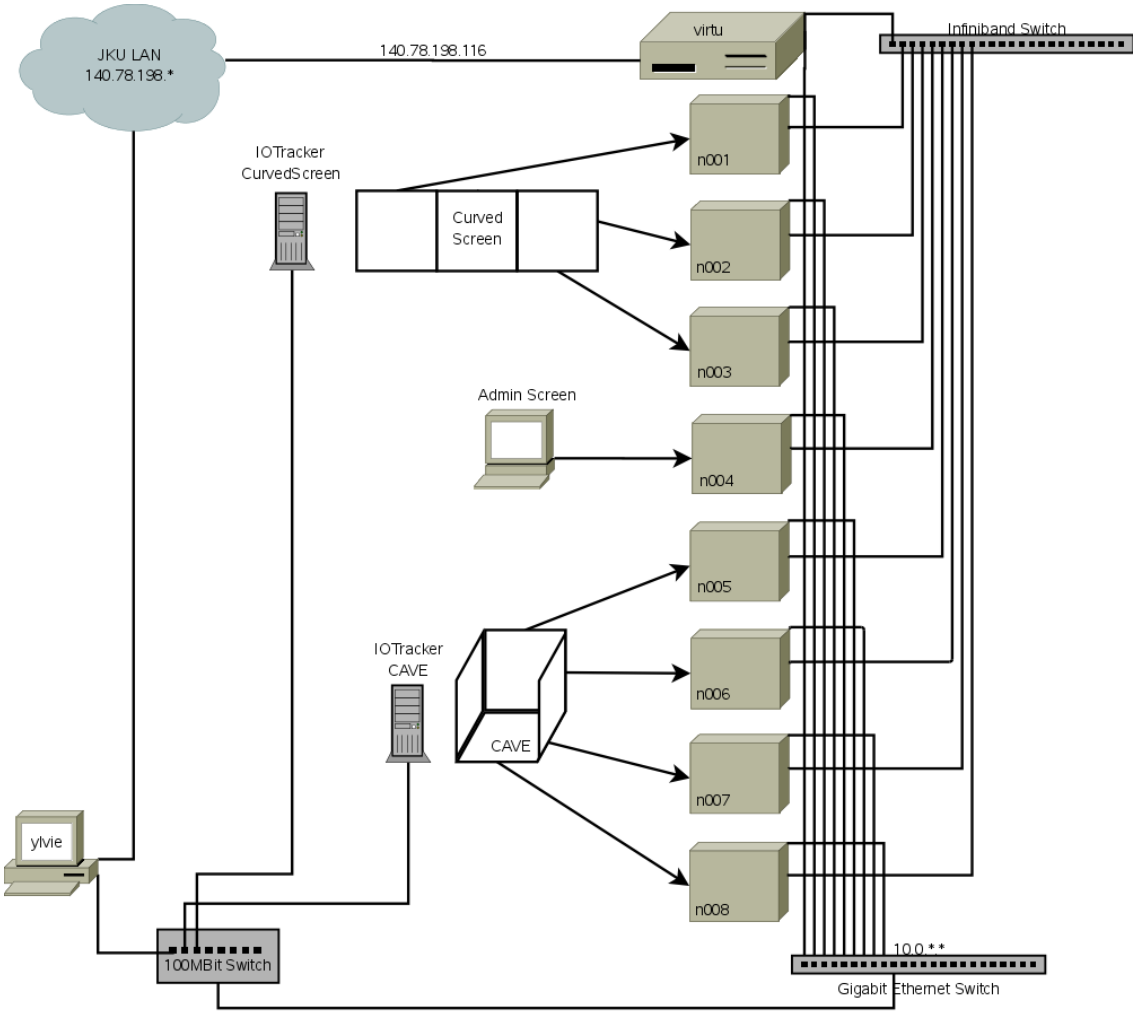

Figure 1: Layout of the Virtu cluster and connections to VRC installations

General information about SGI Virtu systems can be found in [5] (graphics nodes), [4] (head node) and [3] (cluster information).

2.2.1 glxinfo on the graphics nodes

Each of the 8 VN200 graphics nodes is equipped with a *NVidia Quadro FX 5600* graphics card having 1.5 GB of memory. For your convenience, informations about OpenGL capabilities are listed below:

name of display: `:0.0`

display: `:0 screen: 0`

direct rendering: `Yes`

server glx vendor string: NVIDIA Corporation

server glx version string: 1.4

server glx extensions: GLX_EXT_visual_info, GLX_EXT_visual_rating, GLX_SGIX_fbconfig, GLX_SGIX_pbuffer, GLX_SGI_video_sync, GLX_SGI_swap_control, GLX_EXT_texture_from_pixmap, GLX_ARB_create_context, GLX_ARB_create_context_profile, GLX_ARB_multisample, GLX_NV_float_buffer, GLX_ARB_fbconfig_float, GLX_NV_swap_group, GLX_EXT_framebuffer_sRGB, GLX_NV_multisample_coverage, GLX_NV_copy_image, GLX_NV_video_capture

client glx vendor string: NVIDIA Corporation

client glx version string: 1.4

client glx extensions: GLX_ARB_get_proc_address, GLX_ARB_multisample, GLX_EXT_visual_info, GLX_EXT_visual_rating, GLX_EXT_import_context, GLX_SGI_video_sync, GLX_NV_swap_group, GLX_NV_video_out, GLX_SGIX_fbconfig, GLX_SGIX_pbuffer, GLX_SGI_swap_control, GLX_ARB_create_context, GLX_ARB_create_context_profile, GLX_NV_float_buffer, GLX_ARB_fbconfig_float, GLX_EXT_fbconfig_packed_float, GLX_EXT_texture_from_pixmap, GLX_EXT_framebuffer_sRGB, GLX_NV_present_video, GLX_NV_copy_image, GLX_NV_multisample_coverage, GLX_NV_video_capture

GLX extensions: GLX_EXT_visual_info, GLX_EXT_visual_rating, GLX_SGIX_fbconfig, GLX_SGIX_pbuffer, GLX_SGI_video_sync, GLX_SGI_swap_control, GLX_EXT_texture_from_pixmap, GLX_ARB_create_context, GLX_ARB_create_context_profile, GLX_ARB_multisample, GLX_NV_float_buffer, GLX_ARB_fbconfig_float, GLX_NV_swap_group, GLX_EXT_framebuffer_sRGB, GLX_NV_multisample_coverage, GLX_NV_copy_image, GLX_NV_video_capture, GLX_ARB_get_proc_address

OpenGL vendor string: NVIDIA Corporation

OpenGL renderer string: Quadro FX 5600/PCI/SSE2

OpenGL version string: 3.2.0 NVIDIA 190.42

OpenGL extensions: GL_ARB_color_buffer_float, GL_ARB_compatibility, GL_ARB_copy_buffer, GL_ARB_depth_buffer_float, GL_ARB_depth_clamp, GL_ARB_depth_texture, GL_ARB_draw_buffers, GL_ARB_draw_elements_base_vertex, GL_ARB_draw_instanced, GL_ARB_fragment_coord_conventions, GL_ARB_fragment_program, GL_ARB_fragment_program_shadow, GL_ARB_fragment_shader, GL_ARB_framebuffer_object, GL_ARB_framebuffer_sRGB, GL_ARB_geometry_shader4, GL_ARB_half_float_pixel, GL_ARB_half_float_vertex, GL_ARB_imaging, GL_ARB_map_buffer_range, GL_ARB_multisample, GL_ARB_multitexture, GL_ARB_occlusion_query, GL_ARB_pixel_buffer_object, GL_ARB_point_parameters, GL_ARB_point_sprite, GL_ARB_provoking_vertex, GL_ARB_seamless_cube_map, GL_ARB_shader_objects, GL_ARB_shading_language_100, GL_ARB_shadow, GL_ARB_sync, GL_ARB_texture_border_clamp, GL_ARB_texture_buffer_object, GL_ARB_texture_compression, GL_ARB_texture_compression_rgtc, GL_ARB_texture_cube_map, GL_ARB_texture_env_add, GL_ARB_texture_env_combine, GL_ARB_texture_env_crossbar, GL_ARB_texture_env_dot3, GL_ARB_texture_float, GL_ARB_texture_mirrored_repeat, GL_ARB_texture_multisample, GL_ARB_texture_non_power_of_two, GL_ARB_texture_rectangle, GL_ARB_texture_rg, GL_ARB_transpose_matrix, GL_ARB_uniform_buffer_object, GL_ARB_vertex_array_bgra, GL_ARB_vertex_array_object, GL_ARB_vertex_buffer_object, GL_ARB_vertex_program, GL_ARB_vertex_shader, GL_ARB_window_pos, GL_ATI_draw_buffers, GL_ATI_texture_-

float, GL_ATI_texture_mirror_once, GL_S3_s3tc, GL_EXT_texture_env_add, GL_EXT_abgr, GL_EXT_bgra, GL_EXT_bindable_uniform, GL_EXT_blend_color, GL_EXT_blend_equation_separate, GL_EXT_blend_func_separate, GL_EXT_blend_minmax, GL_EXT_blend_subtract, GL_EXT_compiled_vertex_array, GL_EXT_Cg_shader, GL_EXT_depth_bounds_test, GL_EXT_direct_state_access, GL_EXT_draw_buffers2, GL_EXT_draw_instanced, GL_EXT_draw_range_elements, GL_EXT_fog_coord, GL_EXT_framebuffer_blit, GL_EXT_framebuffer_multisample, GL_EXTX_framebuffer_mixed_formats, GL_EXT_framebuffer_object, GL_EXT_framebuffer_sRGB, GL_EXT_geometry_shader4, GL_EXT_gpu_program_parameters, GL_EXT_gpu_shader4, GL_EXT_multi_draw_arrays, GL_EXT_packed_depth_stencil, GL_EXT_packed_float, GL_EXT_packed_pixels, GL_EXT_pixel_buffer_object, GL_EXT_point_parameters, GL_EXT_provoking_vertex, GL_EXT_rescale_normal, GL_EXT_secondary_color, GL_EXT_separate_shader_objects, GL_EXT_separate_specular_color, GL_EXT_shadow_funcs, GL_EXT_stencil_two_side, GL_EXT_stencil_wrap, GL_EXT_texture3D, GL_EXT_texture_array, GL_EXT_texture_buffer_object, GL_EXT_texture_compression_latc, GL_EXT_texture_compression_rgtc, GL_EXT_texture_compression_s3tc, GL_EXT_texture_cube_map, GL_EXT_texture_edge_clamp, GL_EXT_texture_env_combine, GL_EXT_texture_env_dot3, GL_EXT_texture_filter_anisotropic, GL_EXT_texture_integer, GL_EXT_texture_lod, GL_EXT_texture_lod_bias, GL_EXT_texture_mirror_clamp, GL_EXT_texture_object, GL_EXT_texture_shared_exponent, GL_EXT_texture_sRGB, GL_EXT_texture_swizzle, GL_EXT_timer_query, GL_EXT_vertex_array, GL_EXT_vertex_array_bgra, GL_IBM_rasterpos_clip, GL_IBM_texture_mirrored_repeat, GL_KTX_buffer_region, GL_NV_blend_square, GL_NV_conditional_render, GL_NV_copy_depth_to_color, GL_NV_copy_image, GL_NV_depth_buffer_float, GL_NV_depth_clamp, GL_NV_explicit_multisample, GL_NV_fence, GL_NV_float_buffer, GL_NV_fog_distance, GL_NV_fragment_program, GL_NV_fragment_program_option, GL_NV_fragment_program2, GL_NV_framebuffer_multisample_coverage, GL_NV_geometry_shader4, GL_NV_gpu_program4, GL_NV_half_float, GL_NV_light_max_exponent, GL_NV_multisample_coverage, GL_NV_multisample_filter_hint, GL_NV_occlusion_query, GL_NV_packed_depth_stencil, GL_NV_parameter_buffer_object, GL_NV_parameter_buffer_object2, GL_NV_pixel_data_range, GL_NV_point_sprite, GL_NV_primitive_restart, GL_NV_register_combiners, GL_NV_register_combiners2, GL_NV_shader_buffer_load, GL_NV_texgen_reflection, GL_NV_texture_barrier, GL_NV_texture_compression_vtc, GL_NV_texture_env_combine4, GL_NV_texture_expand_normal, GL_NV_texture_rectangle, GL_NV_texture_shader, GL_NV_texture_shader2, GL_NV_texture_shader3, GL_NV_transform_feedback, GL_NV_vertex_array_range, GL_NV_vertex_array_range2, GL_NV_vertex_buffer_unified_memory, GL_NV_vertex_program, GL_NV_vertex_program1_1, GL_NV_vertex_program2, GL_NV_vertex_program2_option, GL_NV_vertex_program3, GL_NVX_conditional_render, GL_SGIS_generate_mipmap, GL_SGIS_texture_lod, GL_SGIX_depth_texture, GL_SGIX_shadow, GL_SUN_slice_accum

2.3 ICatcher Hardware

2.3.1 glxinfo

For your convenience, informations about OpenGL capabilities of the ICatcher system are listed below:

name of display: `:0.0`

display: `:0 screen: 0`

direct rendering: `Yes`

server glx vendor string: `NVIDIA Corporation`

server glx version string: `1.3`

server glx extensions: GLX_EXT_visual_info, GLX_EXT_visual_rating, GLX_SGIX_fbconfig, GLX_SGIX_pbuffer, GLX_SGI_video_sync, GLX_SGI_swap_control, GLX_ARB_multisample, GLX_NV_float_buffer

client glx vendor string: NVIDIA Corporation

client glx version string: 1.3

client glx extensions: GLX_ARB_get_proc_address, GLX_ARB_multisample, GLX_EXT_visual_info, GLX_EXT_visual_rating, GLX_EXT_import_context, GLX_SGI_video_sync, GLX_NV_swap_group, GLX_NV_video_out, GLX_SGIX_fbconfig, GLX_SGIX_pbuffer, GLX_SGI_swap_control, GLX_NV_float_buffer, GLX_ARB_fbconfig_float

GLX extensions: GLX_EXT_visual_info, GLX_EXT_visual_rating, GLX_SGIX_fbconfig, GLX_SGIX_pbuffer, GLX_SGI_video_sync, GLX_SGI_swap_control, GLX_ARB_multisample, GLX_NV_float_buffer, GLX_ARB_get_proc_address

OpenGL vendor string: NVIDIA Corporation

OpenGL renderer string: Quadro FX 3000/AGP/SSE2

OpenGL version string: 2.0.0 NVIDIA 76.76

OpenGL extensions: GL_ARB_depth_texture, GL_ARB_fragment_program, GL_ARB_fragment_program_shadow, GL_ARB_fragment_shader, GL_ARB_half_float_pixel, GL_ARB_imaging, GL_ARB_multisample, GL_ARB_multitexture, GL_ARB_occlusion_query, GL_ARB_point_parameters, GL_ARB_point_sprite, GL_ARB_shadow, GL_ARB_shader_objects, GL_ARB_shading_language_100, GL_ARB_texture_border_clamp, GL_ARB_texture_compression, GL_ARB_texture_cube_map, GL_ARB_texture_env_add, GL_ARB_texture_env_combine, GL_ARB_texture_env_dot3, GL_ARB_texture_mirrored_repeat, GL_ARB_texture_rectangle, GL_ARB_transpose_matrix, GL_ARB_vertex_buffer_object, GL_ARB_vertex_program, GL_ARB_vertex_shader, GL_ARB_window_pos, GL_S3_s3tc, GL_EXT_texture_env_add, GL_EXT_abgr, GL_EXT_bgra, GL_EXT_blend_color, GL_EXT_blend_func_separate, GL_EXT_blend_minmax, GL_EXT_blend_subtract, GL_EXT_compiled_vertex_array, GL_EXT_Cg_shader, GL_EXT_depth_bounds_test, GL_EXT_draw_range_elements, GL_EXT_fog_coord, GL_EXT_framebuffer_object, GL_EXT_multi_draw_arrays, GL_EXT_packed_pixels, GL_EXT_paletted_texture, GL_EXT_pixel_buffer_object, GL_EXT_point_parameters, GL_EXT_rescale_normal, GL_EXT_secondary_color, GL_EXT_separate_specular_color, GL_EXT_shadow_funcs, GL_EXT_shared_texture_palette, GL_EXT_stencil_two_side, GL_EXT_stencil_wrap, GL_EXT_texture3D, GL_EXT_texture_compression_s3tc, GL_EXT_texture_cube_map, GL_EXT_texture_edge_clamp, GL_EXT_texture_env_combine, GL_EXT_texture_env_dot3, GL_EXT_texture_filter_anisotropic, GL_EXT_texture_lod, GL_EXT_texture_lod_bias, GL_EXT_texture_object, GL_EXT_vertex_array, GL_HP_occlusion_test, GL_IBM_rasterpos_clip, GL_IBM_texture_mirrored_repeat, GL_KTX_buffer_region, GL_NV_blend_square, GL_NV_copy_depth_to_color, GL_NV_depth_clamp, GL_NV_fence, GL_NV_float_buffer, GL_NV_fog_distance, GL_NV_fragment_program, GL_NV_fragment_program_option, GL_NV_half_float, GL_NV_light_max_exponent, GL_NV_multisample_filter_hint, GL_NV_occlusion_query, GL_NV_packed_depth_stencil, GL_NV_pixel_data_range, GL_NV_point_sprite, GL_NV_primitive_restart, GL_NV_register_combiners, GL_NV_register_combiners2, GL_NV_texgen_reflection, GL_NV_texture_compression_vtc, GL_NV_texture_env_combine4, GL_NV_texture_expand_normal, GL_NV_texture_rectangle, GL_NV_texture_shader, GL_NV_texture_shader2, GL_NV_texture_shader3, GL_NV_vertex_array_range, GL_NV_vertex_array_range2, GL_NV_vertex_program, GL_NV_vertex_program1_1, GL_NV_vertex_program2, GL_NV_vertex_program2_option, GL_SGIS_generate_mipmap, GL_SGIS_texture_lod, GL_SGIX_depth_texture, GL_SGIX_shadow, GL_SUN_slice_accum

glu version: `1.3`

glu extensions: `GLU_EXT_nurbs_tessellator, GLU_EXT_object_space_tess`

3 Useful commands

3.1 Environment modules – module

`module` [1] can dynamically add, modify and remove environment variables of the shell and supports all major shells (such as bash, tcsh, ksh, and even perl). On virtu all software packages below `/jkuvrc/packages` are accessible via the `module` command. For each package version, a so called *modulefile* exists.

A typical modulefile of a software package would add the package's `bin` directory to the `PATH`, its `lib` directory to the `LD_LIBRARY_PATH` and set an environment variable `PACKAGENAME_HOME` to the package base directory.

3.1.1 Usage

Table 3 shows lists possible module invocations.

Module invocation	Description
module avail	List all available modules and their versions.
module help	General usage information about modules.
module help PACKAGE	Print help-text for the given module. Some modules also use this to give tips and hints about their usage.
module initlist	Show a list of modules which are autoloaded.
module initadd PACKAGE	Autoload the given module at login.
module initrm PACKAGE	Remove the given module from the list of autoloaded modules.
module list	List all currently loaded modules. Note: This should at least show the *null* module.
module load PACKAGE	Load the given module (some modules will autoload other modules)
module show PACKAGE	Show a summary of how the given module would alter your environment when loaded.

Table 3: The module command

3.1.2 Using modules in shell scripts

The basic way of using modules has 2 steps:

1. Load the modules needed by a program
2. Execute the program

This approach works fine. However, once you try to automate you program execution you will probably try putting it into a shell script similiar to the one in table 4.

Once you execute that script, your shell will cough up an error message about not finding the `module` command. The reason for this is the way module interacts with your shell. The `module` command is not an executable file, but rather a shell function. Shell functions are not inherited by subshells, and the `module` shell function is only declared for your login shell.

In order to use the `module` command in a shell script, you have to manually load it as shown in table 5.

```
#!/bin/sh

# /bin/sh complains:
# "module: command not found"
module list

# however, you still have the same environment set up as the parent shell:
echo $LOADEDMODULES

# ... success of your program depends on whether you set up the correct
# environment before calling this script!
```

Table 4: Using the module command in a shell script the wrong way

```
#!/bin/sh

# enable the module command:
. /usr/share/modules/init/`basename $SHELL`

# use the module command:
module list

# module load ...

# ... execute some program ...
```

Table 5: Using the module command in a shell script the right way

4 Tracking Systems

4.1 Overview

Both CAVE and CurvedScreen use an *IOTracker* optical tracking system. The IOTracker PCs send the tracking data to virtu using one-way UDP communication. As soon as the projectors of CAVE or CurvedScreen are turned on, the respective IOTracker system is also turned on. Normally, it takes about 2 minutes until the tracking system has been started and is usable. During this time, you may already start a program which uses tracking – the tracking server software on virtu is always available, even when the tracking hardware doesn't deliver any data.

Currently, three software stacks are available on virtu: CAVELib[2], trackdAPI, and VRPN[6].

CAVELib is a complete API for creating VR-applications, while trackdAPI only provides an interface to read tracking data from the trackd server. The trackd server runs on the head-node (i.e. virtu) and receives tracking data from both IOTracker systems. CAVELib and trackdAPI both use the same shared-memory communication model to connect to the *trackd* server running on virtu. Therefore, a program using trackd must run on virtu itself, not on a cluster node!

VRPN is a tracking library using TCP or UDP connections between server and client. There is one VRPN server running on virtu, which receives exactly the same tracking data from the IOTracker systems as trackd does. Another VRPN server runs on host *ylvie-ge*, which is used to connect the Nintendo Wii Remotes.

4.2 Devices

Table 6 shows the input devices and trackers, which are available at the VRC using trackd or VRPN. The trackd columns list values for both tracker (for position and orientation) and controller (for buttons and valuators) in the format *tracker*/*controller*. The following sections will describe the various input devices in detail.

Device	Trackd SHM-key	Trackd device index	VRPN device name	VRPN device Index
		CAVE		
Head-target	4129/-	0/-	cavetracker@virtu	1
IOPad	4129/4128	1/0	cavetracker@virtu	2
T6001	4129/4128	6/-	cavetracker@virtu	7
T6002	4129/4128	7/-	cavetracker@virtu	8
T6003 (IOPen)	4129/4128	2/1	cavetracker@virtu	3
T6004 (WiiMote)	4129/-	3/-	cavetracker@virtu	4
T6005	4129/4128	4/-	cavetracker@virtu	5
T6006	4129/4128	5/-	cavetracker@virtu	6
		CurvedScreen		
Head-target	4125/4127	0/-	curvedscreentracker@virtu	1
IOPad	4125/4127	1/0	curvedscreentracker@virtu	2
T6001	4125/-	6/-	curvedscreentracker@virtu	7
T6002	4125/-	7/-	curvedscreentracker@virtu	8
T6003 (IOPen)	4125/4127	2/1	curvedscreentracker@virtu	3
T6004 (WiiMote)	4125/-	3/-	curvedscreentracker@virtu	4
T6005	4125/-	4/-	curvedscreentracker@virtu	5
T6006	4125/-	5/-	curvedscreentracker@virtu	6
		Other		
WiiMote buttons	-/-	-/-	wiimote0@ylvie-ge	1

Table 6: Overview of VRC trackers and input devices

4.2.1 IOPad

The IOPad is a modified Logitech Cordless Rumblepad 2, fitted with optical markers for tracking. Two IOPads are available at the VRC, one for the CAVE and one for the CurvedScreen. The button layout of the IOPad is shown in figure 2. Only the buttons and valuators (joysticks) which are highlighted are available for input.

The IOPad is the standard input device for CAVELib applications.

4.2.2 IOPen

The IOPen is a modified commodity presenter with two buttons and a laser-pointer, fitted with a IOTracker standard target (T6003). Figure 3 shows the button layout of the IOPen. Only one button can be active at any time (this is a hardware constraint).

4.2.3 WiiMote

We have two Nintendo Wii Remote (aka "WiiMote") controllers at the VRC, along with two Nunchucks. One of the Wii Remotes has been fitted with an IOTracker standard target (T6004) to allow optical tracking.

Although the standard target on the Wii Remote can be tracked using trackd, it is only possible to access the button (and joystick) data using VRPN.

To use a Wii Remote controller, press buttons "1" and "2" simultaneously. This turns on the remote and makes it discoverable to the VRPN server on ylvie. Once a connection has been established, the blinking led is replaced by a steady lit led and the rumble is turned on shortly. Button and valuator events are now available to the client. Figure 4 shows the button layout of the Wii Remote when used via VRPN.

Important: Please remember to turn off the remote after use! You can turn off the Wii Remote by pressing and holding the power button for several seconds (until the status led turns off).

Figure 2: IOPad with usable controls highlighted

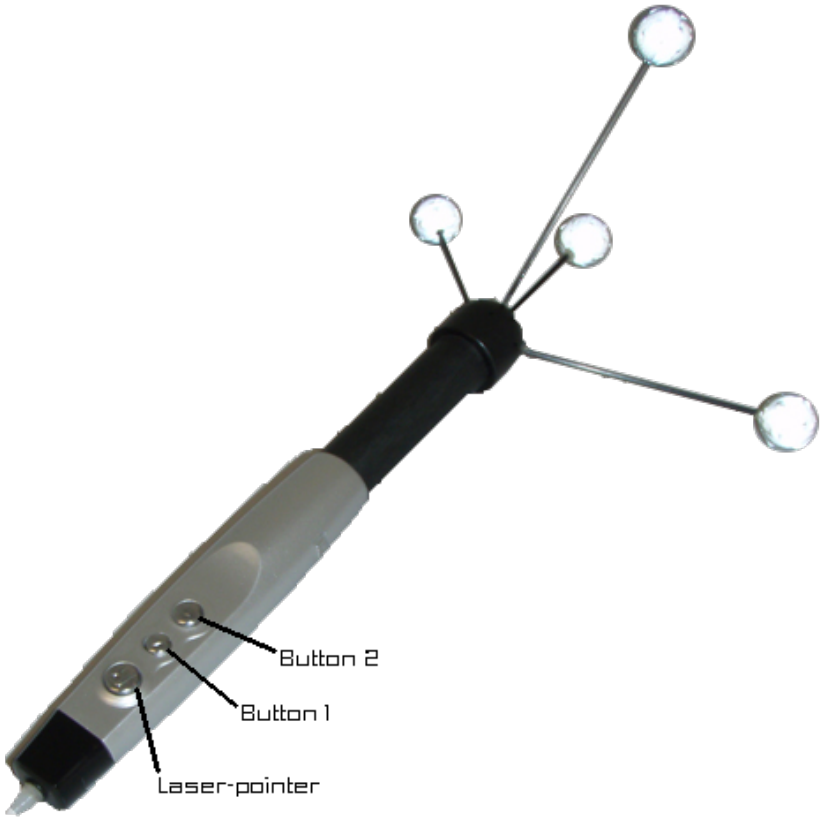

Figure 3: IOPen with highlighted buttons

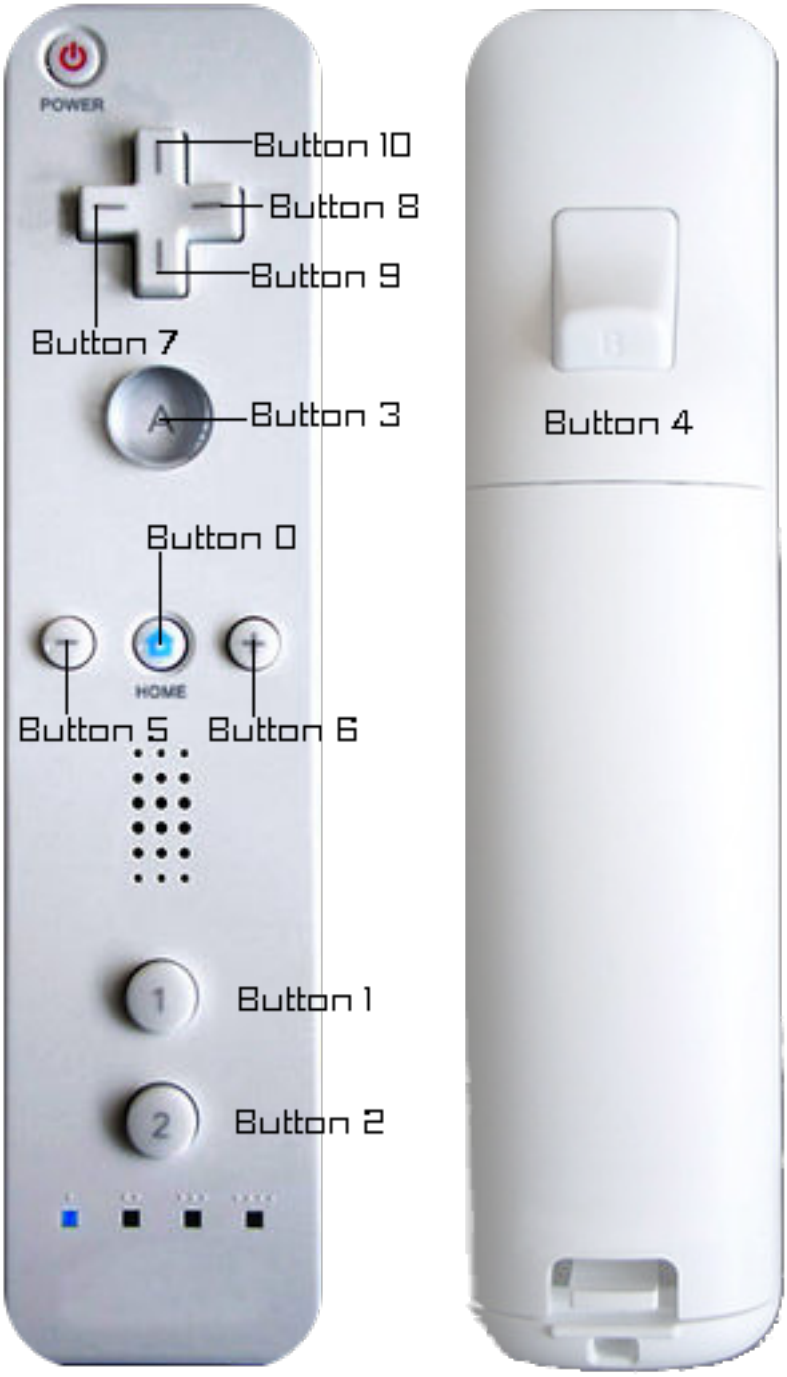

Figure 4: Wii Remote button enumeration with VRPN

5 Tips & Tricks

5.1 Debugging

5.1.1 Various debugging commands

- ddd
  This is a graphical debugger (i.e. a frontend to `gdb`).
- insight
  This is another graphical debugger. Use "`module load debugger/insight`" to make it available.
- ldd
  Show runtime DSO-dependencies.
- nm
  Print a name list of an object file. Useful examples: Just inspect the symbol-table:

  ```
  user@virtu:~> nm myobj.o | less
  ```

  Search for a specific function/object:

  ```
  user@virtu:~> nm mylib.so | grep "functionname"
  ```

  Display all globally bound functions:

  ```
  user@virtu:~> nm mylib.so | grep 'FUNC |GLOB'
  ```

  You can tell nm to print human-readable names with the "`-C`" option.

Note: make sure your LD_LIBRARY_PATH is set to the same value as during program-execution, or you'll get false results. This also means that you should load the same modulefiles (section 3.1 that you normally use when starting your program.

5.1.2 Showing the framerate(FPS)

In GLUT-based applications, you can print the framerate to standard output every X milliseconds by setting the `GLUT_FPS` environment variable. To print the framerate every 5 seconds use:

```
user@virtu:~> export GLUT_FPS=5000
```

5.2 Tools

- nedit
  A lightweight X-based editor featuring syntax higlighting.
- kate
  A KDE-based multi-document editor.
- konqueror
  The KDE filemanager and webbrowser.
- firefox
  The Mozilla-Firefox browser.
- w
  Show, which users are logged in (and how they logged in).
- kpdf
  A pdf-viewer.

Tip: Instead of typing "`exit`", you can also press < `CTRL` > +`D` (the `EOF`-character). Instead of typing "`clear`", you can use < `CTRL` > +`L`.
If you typed a long or complicated command, but then decide that you don't want to execute it yet, you can use < `ALT` > +# to comment it out and save it to the command history. Afterwards you can access it by using the up-arrow key.

6 Troubleshooting

6.1 General problems

6.1.1 SSH connection problems

What happened: You need to supply a password every time you connect from the head-node to one of the graphics nodes, or a script that executes commands on one of the graphics nodes fails.

What it means: Most of the time this means that your account wasn't configured to public-key based authentication with the cluster-nodes. To get an idea of what went wrong, you can execute the script ssh-setup.sh:

```
user@virtu:~> ssh-setup.sh
Checking availability of head-node host key:      [  OK  ]
Checking existance of public-key file:            [  OK  ]
Checking two-way connection to node n004          [  OK  ]
```

6.2 CMake errors

6.2.1 Library not found

What happened: You tried to configure a project using cmake, but cmake came up with an error like this one:

```
user@virtu:~> cmake ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
CMake Error at /usr/local/share/cmake-2.6/Modules/
    FindPackageHandleStandardArgs.cmake:57 (MESSAGE):
  Could NOT find inVRs (missing: inVRs_LIBRARY inVRs_INCLUDE_DIR)
  Call Stack (most recent call first):
    cmake/Modules/FindinVRs.cmake:164 (
        FIND_PACKAGE_HANDLE_STANDARD_ARGS)
           cmake/config.cmake:18 (find_package)
             CMakeLists.txt:58 (include)


                -- Configuring incomplete, errors occurred!
```

What it means: The cmake module `cmake/Modules/FindinVRs.cmake` did not find its library package based on environment variables or standard locations. Normally, this means that you didn't load the appropriate modulefile (see section 3.1).

In this case, loading the modulefile will set the neccessary environment variables, so that cmake can automatically find the library:

```
user@virtu:~> module load inVRs
user@virtu:~> cmake ..
-- Found inVRs: /jkuvrc/packages/inVRs/inVRs-v1.0alpha5-svn2086/lib/
    libinVRsSystemCore.so
-- Configuring done
-- Generating done
-- Build files have been written to: /home/edvz/zing/examples/
    graphics_VR/inVRs/GoingImmersive/build
```

6.3 Compiler and linker errors

6.3.1 Undefined reference to `__gxx_personality_v0`

This error occurs, if you use gcc instead of g++ to compile a C++ source file.

6.3.2 `GL/glut.h`: No such file or directory

This happens when compiling on the graphics nodes instead of `virtu` (i.e. the head-node) itself. Connect to the head-node and try again.

6.4 Runtime errors

6.4.1 Error while loading shared libraries

What happened: Your program does not start because of linker errors:

```
user@virtu:~/examples/graphics_VR/inVRs/GoingImmersive/build> ./
    GoingImmersive
./GoingImmersive: error while loading shared libraries: libavatara.so:
    cannot open shared object file: No such file or directory
```

What it means: The runtime linker could not resolve all of the dynamic libraries requested by the program. To get more detailed information about the missing dependencies, use `ldd`:

```
user@virtu:~> ldd ./GoingImmersive |grep 'not found'
        libavatara.so => not found
        libinVRsControllerManager.so => not found
        libCAVESceneManager.so => not found
        libtrackdAPI_CC.so => not found
```

In most cases, the missing dependencies are caused by non-loaded modulefiles (see section 3.1). If you have no idea, which modulefile is needed for a given `.so` file, you can try to locate it:

```
user@virtu:~> locate libavatara.so |grep /jkuvrc/packages
/jkuvrc/packages/inVRs/DIST/inVRs-v1.0alpha5-svn2086/build-with-vrpn/
    external/avatara-1.0/libavatara.so
/jkuvrc/packages/inVRs/DIST/inVRs-v1.0alpha5-svn2086/build/external/
    avatara-1.0/libavatara.so
/jkuvrc/packages/inVRs/inVRs-v1.0alpha5-svn2086/lib/libavatara.so
```

In this example, one can see that `libavatara.so` belongs to the *inVRs* package. Systematically load the appropriate modulefiles until `ldd` shows no missing dependencies:

```
user@virtu:~> module load inVRs
user@virtu:~> ldd ./GoingImmersive |grep 'not found'
user@virtu:~> ./GoingImmersive
```

If your program uses a library for which there is no modulefile, you can simply add its path to the `LD_LIBRARY_PATH` environment variable:

```
user@virtu:~> ldd ./myProgram |grep 'not found'
        libavatara.so => not found
        libinVRsControllerManager.so => not found
        libCAVESceneManager.so => not found
        libtrackdAPI_CC.so => not found
        libMyAdditionalSpecialLibrary.so => not found
user@virtu:~> module load inVRs
user@virtu:~> ldd ./myProgram |grep 'not found'
        libMyAdditionalSpecialLibrary.so => not found
user@virtu:~> echo $LD_LIBRARY_PATH
/jkuvrc/packages/vrpn/vrpn-7.21/lib:/jkuvrc/packages/trackdAPI//lib64/
    linux_x86_64/:/jkuvrc/packages/OpenAL/OpenAL-1.8.466/lib:/jkuvrc/
    packages/ODE/ode-0.8/lib:/jkuvrc/packages/OpenSG/OpenSG
    -1.8.0-2009-07-27-ode0.8/lib/opt:/jkuvrc/packages/inVRs/inVRs-v1.0
    alpha5-svn2086/lib:/opt/sgi/islecm/java/lib/amd64:/opt/sgi/islecm/
    java/lib/amd64/server:/usr/lib64/mpi/gcc/openmpi/lib64:
user@virtu:~> export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/path/to/my/
    additional/special/library"
user@virtu:~> ldd ./myProgram |grep 'not found'
user@virtu:~> ./myProgram
```

6.4.2 Trackd errors

What happened: Tracking is working, but the joystick on the IO-Pad doesn't work.

What this means: The IO-Pad goes into energy-saving-mode after a couple of minutes without a key press. Push any button and the joystick should start working again.

What happened: Tracking is working, but neither the joystick nor the buttons on the IO-Pad work.

What this means: If the led besides the *mode* button is turned on, push the *mode* button again to set the IO-Pad into normal mode. Otherwise, the batteries might be empty.

What happened: Your program runs without obvious errors, but no tracking data is received.

What happened: The tracking server is not (yet) ready. The tracking servers for are started as soon as the projectors for CAVE and CurvedScreen are turned on. The average time needed to boot the servers is about 3 minutes. During this time, no tracking data is generated.

If for some reason the tracking server did not receive a wake-on-lan signal, you can start them using the "start tracking hardware" button on the Crestron panel.

What happened: Your program uses trackd and aborts with the message "`ERROR: can't get tracker shared memory!`".

What this means: Your program cannot communicate with `trackd`. This can have three reasons: either you are running the program on the *wrong node*, or the tracker/controller *shared memory key is wrong*, or ***`trackd` is not running*** correctly.

It is easy to forget connecting to the head-node before running your program, especially if you are working locally on the terminal on `n004`. Shared-memory communication simply cannot work between different nodes!

To find out the correct shared memory keys for accessing the tracker, consult the help for the trackdAPI modulefile:

```
user@virtu:~> module help trackdAPI

---------- Module Specific Help for 'trackdAPI/2.2' ----------------

trackdAPI - prepares your environment for using the trackdAPI package.
trackdAPI allows non-CAVELib programs to interface with a trackd
    server

Shared memory segments used at the JKU VRC are:
 Key   | Description
 ------+-----------------------------
 4125  | Tracker data CurvedScreen
 4127  | Button data CurvedScreen
 4128  | Button data CAVE
 4129  | Tracker data CAVE

A useful utility for testing trackd is read-trackd. For CurvedScreen,
    use:
 read-trackd -trackerDaemonKey 4125 -controllerDaemonKey 4127
For CAVE use:
 read-trackd -trackerDaemonKey 4129 -controllerDaemonKey 4128

Find out more about trackdAPI and trackd at:
 http://www.mechdyne.com/integratedsolutions/software/products/trackd/
     trackd.htm
```

If your program already uses the correct keys, you should check if `trackd` is running:

```
user@virtu:~> ps -u tracking
  PID TTY          TIME CMD
 4887 ?        00:00:00 lmgrd
 4888 ?        00:00:00 VRCO
 5351 ?        03:16:33 trackd
```

If it is running, there should also be 4 shared memory segments owned by user `tracking`:

```
user@virtu:~> ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch
    status
0x00001021 393216     tracking   666        1308       1
0x00001020 425985     tracking   666        296        1
0x0000101d 458754     tracking   666        1308       1
0x0000101f 491523     tracking   666        296        1
```

If trackd doesn't run correctly, use the button on the Crestron Panel to restart trackd.

6.5 Sound programming

- Use the right sampling rate!
  If your soundfile does play on other computers, but not on virtu, try playing it using "aplay". If you can hear it now, the file should be correct.

- Is your soundserver running?
  If you use bergen or VSS, make sure the sound-server is running. For bergen, start `snerd`. for VSS, use `vss`.

- Is the soundcard used by another program?
  You can get a list of all programs using the soundcard by issuing the following command:

```
user@virtu:~> lsof /dev/snd/*
user@virtu:~> lsof /dev/dsp*
```

  If someone is using the sound device, you will get something like the following:

```
COMMAND   PID  USER  FD   TYPE DEVICE SIZE      NODE NAME
aplay   25495 vradm mem    CHR 116,16      184549613 /dev/snd/pcmC0D0p
```

  Then you can terminate the process via `kill` *`PID`*. Often, the program blocking the soundcard is `artsd`. If a process of another user is blocking the sound-device, you can kill the process via the respective button on the Crestron remote-control on page *Admin → Audio/-Page 2*.

- Soundtest
  You can test if the sound is working on the page *Admin → Audio/Page 2* of the Crestron remote-control.

- Note: The speakers in VRC are normally only turned on, when either Curved Screen or CAVE (or both) are running. You can use the Crestron remote-control to switch the amplifiers independently via *Admin → Audio*.

References

[1] Peter W. Osel John L. Furlani. *Homepage of the Environment Modules Project*. Available from World Wide Web: `http://modules.sourceforge.net/`.

[2] Mechdyne. *CAVELib User Guide*, Feb. 2004. Available from World Wide Web: `http://www.mechdyne.com/integratedSolutions/software/products/CAVELib/HELP/index.html`.

[3] SGI. *SGI Altix XE1300 Cluster Quick Reference Guide*, Feb. 2008. Available from World Wide Web: `http://techpubs.sgi.com/library/tpl/cgi-bin/summary.cgi?coll=hdwr&db=bks&docnumber=007-4979-004`. publication number 007-4979-00x.

[4] SGI. *SGI Altix XE250 System User's Guide*, Jan. 2009. Available from World Wide Web: `http://techpubs.sgi.com/library/tpl/cgi-bin/summary.cgi?coll=hdwr&db=bks&docnumber=007-5467-004`. publication number 007-5467-00x.

[5] SGI. *SGI Virtu VN200 Graphics Node Hardware User's Guide*, Apr. 2009. Available from World Wide Web: `http://techpubs.sgi.com/library/tpl/cgi-bin/summary.cgi?coll=hdwr&db=bks&docnumber=007-5483-001`. publication number 007-5483-00x.

[6] University of North Carolina at Chapel Hill. *Homepage of the Virtual Reality Peripheral Network (VRPN) project*. Available from World Wide Web: `http://www.cs.unc.edu/Research/vrpn/`.