

# **VR Juggler**

## **Configuration Guide**

---

# VR Juggler: Configuration Guide

Published \$Date: 2002/03/19 00:25:02 \$

---

---

---

# Table of Contents

Preface .....	vi
1. Introduction .....	1
An overview of VR Juggler configuration .....	1
2. Building a configuration file from scratch .....	2
Before you start: collecting information .....	2
Create a new configuration file .....	2
Decide on the origin .....	3
Configure trackers .....	3
Create position proxies .....	3
Creating Proxy Aliases (optional) .....	4
Configure other input devices .....	4
Creating Digital and Analog Proxies .....	5
Add User configuration .....	5
Add Displays .....	5
Create a DisplaySystem chunk .....	5
Define display windows .....	6
Additional features .....	7
Configuring Performer API features .....	7
Configuring audio features .....	7
Enabling dynamic reconfiguration .....	7
Logging performance data .....	7
3. Configuring other hardware devices .....	9
Desk or wall-based projection displays .....	9
Head-mounted displays .....	9
Using the Trackd tracker daemon with VR Juggler .....	9
Using InterSense trackers .....	10
Using Fakespace Pinch Gloves .....	11
4. Configuring simulated devices .....	12
Creating a keyboard input window .....	12
Creating simulated devices .....	12

---

## List of Figures

1.1. VjControl's config file editing panel. ....	
2.1. Editor window for MotionStar input driver. ....	
2.2. The editor window for a Position Proxy called "VJHead". The proxy attaches to a device named "Motion# Star", and applies a transformation and rotation to the device data. ....	
2.3. Configuration for the Immersion Ibox. ....	
2.4. User data ConfigChunk. ....	
2.5. DisplaySystem chunk. ....	
2.6. Surface Display chunk for the front screen. ....	
2.7. ConfigChunk for activating performance measurements. ....	
2.8. Configuring a log file for VR Juggler performance data. ....	
3.1. ConfigChunks for the Trackd input device daemon. ....	
3.2. Configuration for InterSense tracking system. ....	10
4.1. Configuration of simulated devices. These are the digital and positional simulators used to provide the user's head control and wand buttons in the simstandalone.config file included in the VR Juggler distribution. .....	13

---

# Preface

This book describes how to create configuration files for VR Juggler. It discusses many of the details of configuring a VR System from scratch.

This book assumes that the user has read the VR Juggler Getting Started Guide. The VjControl User's Guide may also prove helpful.

---

# Chapter 1. Introduction

Configuring software for an immersive VR system can be a difficult and time consuming process. This is largely because of the size and complexity of the hardware components that make up the system - there may be multiple displays and a multitude of input devices. Furthermore, many systems are unique, or at least highly customized.

VR Juggler attempts to make the configuration process as simple as possible. It includes a GUI application called VjControl to edit config files. Use of the GUI has many advantages: it can provide an organized view of the configuration information, with help information or suggested values readily available. It can prevent many simple mistakes (such as typos in text-based configuration files) and help resolve others.

In addition to this, the VR Juggler config files themselves are well-structured, allowing the configurations of various elements of the VR system to be created separately and then combined in various ways.

## An overview of VR Juggler configuration

VR Juggler's configuration files are composed of units called ConfigChunks. A ConfigChunk contains the configuration information for a single element of VR Juggler. These elements might be display windows, or device drivers, or even applications. Naturally, the ConfigChunk for a display window contains very different information from that of a device driver.

All ConfigChunks have individual names, so that multiple chunks for the same kind of element can be distinguished. This means, for example, that a VR Juggler configuration can include several display windows with separate screen coordinates and viewing parameters, each with a unique name. It also means that VR Juggler can use an arbitrary number and variety of displays or devices at any time, limited only by the capabilities of the underlying hardware.

VjControl represents configuration files as a tree of ConfigChunks, as seen in Figure 1.1. The tree structure exists to help organize the contents of the file, and is user-customizable. VjControl also allows users to view multiple configuration files side-by-side, making it easy to copy ConfigChunks from one to another.

---

# Chapter 2. Building a configuration file from scratch

VR Juggler includes a number of sample configuration files for common VR systems. These files are based on the systems readily available to the VR Juggler developers and our contributors, but obviously we cannot provide a comprehensive set of example configurations - the field is simply too diverse.

In this section we will demonstrate how to create a VR Juggler configuration file from scratch, describing each element and suggesting an order of steps to setup a new VR system in the easiest possible way. Our example system will be a four-walled projection system similar to Iowa State University's C4 device. After the step-by-step example, we will also briefly describe major variations, such as the differences necessary to create a configuration for a Head-Mounted Display, and some specific examples of setting up various input devices.

## Before you start: collecting information

Configuring a VR system is much easier if all the necessary information is readily at hand. Sometimes, though, it can be hard to figure out exactly what is needed. The following non-exhaustive list suggests some of the data that it may be helpful to collect:

- **Display Information.** This includes knowing where your displays are in space. In projection-based VR systems, this will include the positions of each screen. For a head-mounted display, it would be the positions and sizes of the display surfaces relative to the positions of the users' eyes. This also includes knowing how the displays of the host computer map to the displays of the VR system - for example, the position and size of a graphics window needed to drive a given VR display.
- **Input Device information.** This includes knowing what input devices you intend to use and how they are connected to the host computer. For some devices, this will be a serial port name and baud rate, for others it might be a network address and port number. Tracking systems, in particular, tend to have a large number of device-specific configuration options. For any tracking system, you will need to know the coordinate system its data uses.
- **Alternate configurations.** Some VR systems can be used in a variety of ways. For example, you might have a desk with a movable display surface, or you may need to choose between mono and stereo display modes, or you may have a head-mounted display that supports multiple video aspect ratios. Since VR Juggler applications can load multiple configuration files, it may be helpful to create a "base" config file with the information that never changes, as well as several additional helper files that describe the components of the system that may be reconfigurable.
- **VjControl usage information.** The VjControl program used to edit configuration files has many powerful capabilities. While it is designed to be easy to learn and use, it may be helpful to browse the *VjControl User's Guide*. The information in the Guide is also available via VjControl's help menu.

## Create a new configuration file

Many of the sample configuration files included with VR Juggler are modular - that is, different parts of the system are described in different files, and multiple files are loaded to configure the entire VR system. To simplify this example, we'll put all of our configuration information into a single file.

First, start VjControl and click on the Configuration tab. The configuration panel is split into two side-by-side panels. We will only be using one side for this tutorial. Click on New to create a new file. If you want, click Save to give your file a name.

## Decide on the origin

All displays and input data in VR Juggler are specified relative to a single origin point. The origin is some point in the physical space inside or around the VR System. Before we configure anything else, we have to decide where that point will be.

For a Surround-Screen VR (SSVR) system, we decided that the origin point would be the center of the floor, equidistant from each wall. This is a common convention and is usually very convenient.

We also need to define the orientation of our origin. Again there is a common convention which we recommend. We use a right-handed coordinate system, with positive  $X$  to the right of an observer looking toward the front screen. Positive  $Y$  is up and positive  $Z$  is toward the rear of the system (away from the front screen).

Note that in the sample VR Juggler configuration files, all positions related to displays or input devices are given in feet.

## Configure trackers

The easiest place to start is at the bottom, with out input hardware. Open up the Input folder, and then Input Devices. You will see a list of all the device drivers we can create configurations for. For this example, we will use an Ascension MotionStar for our tracking system. Right-click on the MotionStar label in the tree, and select Insert from the menu. An empty ConfigChunk describing the MotionStar driver is created.

Now we need to edit the MotionStar configuration to describe the details of our system. Double-clicking on the newly-created MotionStar ConfigChunk will open an editor window. This window shows the complete set of configuration options for VR Juggler's MotionStar driver.

The MotionStar is a separate computer that sits on a network and provides position data to other interested hosts. So it should not be surprising that the first few properties in the configuration are networking settings - IP address and port of the MotionStar, for example. Other input devices may use other communications methods, such as a serial connection. In that case, the configuration for that device would include port name and baud rate settings instead of the networking settings.

The Translation and Rotation properties are common to all positional devices. They describe the position of the tracking system's origin relative to the VR Juggler origin we selected in the previous step. In our example, the tracking system's origin is the position and orientation of its transmitter, which is suspended above the floor of our SSVR system. The  $Y$  value for our translation is therefore about +8.3 (for a transmitter 8.3 feet above the VR Juggler origin), and the  $X$  and  $Z$  values are the slight offset of the transmitter from the center of the floor. The rotation values (in degrees) represent the way the transmitter is facing.

Most of the other configuration options shown for the MotionStar are particular to its specific hardware and low-level software driver, and are described by the device's own documentation. However, we should draw attention to the Number of Birds property. The MotionStar hardware supports multiple positional sensors, called birds. In our example setup, we have two sensors. One is attached to a pair of stereo glasses, and the other is mounted to the wand which enables users to interact with the virtual environment.

Note that there is nothing limiting us to having a single tracking system, or even a single MotionStar driver. For example, we could create two MotionStar configurations in the same file, each with different network settings, each talking to a different physical MotionStar. The Instance Name at the top of the configuration window can be used to distinguish one from the other.

## Create position proxies

In VR Juggler, we generally do not interact with input devices directly. Instead, we create proxies for them. A Position Proxy represents a single positional input. It is used to provide a simple, generic interface to position data, abstracting away from the actual hardware.

As mentioned previously, the MotionStar in our example setup has two sensors, so we will need to create two Position Proxies. Expand the Proxies and Aliases subtree under Input in the configuration panel. Right click on Position Proxy and insert a new instance (and repeat to create the second proxy).

Now it is time to fill in the particular information for each Proxy. We can start with the instance names, calling one proxy "VJHead" and the other "VJWand". VR Juggler applications use these proxy names to access input devices, and these particular choices of names are a convention used by most VR Juggler applications.

Each proxy has a pulldown menu where we can select the physical device it maps to - these will both refer to the MotionStar driver instance we've already configured. There is also a Unit option, which tells us which of the individual position inputs of the device to use. Our driver is configured for two sensors, so one of our proxies will be unit 0 and the other will be unit 1.

The Position Proxy also has translation and rotation options. This is useful for transforming the raw position data into something that is easy to use and manipulate. For example, in our particular example VR system, the physical tracking sensor that we are using as "VJHead" is mounted sideways on the earpiece of a pair of stereo glasses. The transformation settings we use (as seen in the screenshot) change that data so that the position returned to VR Juggler and the application itself is the position of a point directly between the user's eyes.

## Creating Proxy Aliases (optional)

Proxy Aliases provide a way to introduce multiple names for a single Proxy. A Proxy Alias ConfigChunk contains only two properties: its name and the name of the Proxy it points at.

The Proxy Alias provides a convenient way to reassign a commonly-used name like "VJHead" or "VJButton0" from one proxy to another. While not required, use of Proxy Aliases can make configuration files easier to modify and administrate.

## Configure other input devices

In addition to moving around within an environment, users will also want to interact with the application in some way. There are many sorts of devices that can be used in a VR system. Wands and joysticks of various sorts are very popular. These are simple handheld devices, usually with a position tracking sensor attached to them, and with a number of buttons that the user can press to send commands to an application (such as "fly forward" or "pick up the object I'm touching"). Gloves that sense the user's hand movements are also popular (see the section called "Using Fakespace Pinch Gloves", below, for an example). In surround-screen VR systems, it is also possible to bring physical devices - for example, a bicycle or the cab of a truck - into the VR system. In this case, the user can interact with the application by manipulating pedals, handlebars, or steering wheels.

In VR Juggler, we divide most of these devices into digital or analog inputs. Digital devices are buttons, like on a mouse or joystick, and simply have on and off values. Analog devices return a range of data, and can be used for steering wheels or foot pedals, among other things.

Several tracking system manufacturers include handheld wand-style devices that can be used as a regular sensor in their hardware. For example, see the section called "Using InterSense trackers" at the end of this article for information about using digital inputs supplied by an InterSense tracking system.

At VRAC, we have hooked up a large variety of custom devices via the IBox, a simple input device manufactured by Immersion Corp. The IBox connects to the serial port of a computer, and provides four digital inputs and four analog inputs. See Figure 2.3 for an IBox ConfigChunk.

It is also possible to use simulated devices to achieve the same functionality. One of the wands in use at VRAC is in fact a wireless 2D mouse attached to a spare computer. We can configure VR Juggler applications to open a simu#

---

In VR Juggler 1.0, the Proxy Alias chunk actually contains an AliasName property in addition to the actual chunk name. This redundancy is likely to be removed in the future. For now, it is recommended that users always put the same string in the Alias chunk's Name and AliasName.

lator input window on the spare computer (using an exported X Windows display) and map the wireless mouse's buttons to a simulated digital device. This is a fairly extreme example of the flexibility provided by VR Juggler's input system.

## Creating Digital and Analog Proxies

As with tracking devices, VR Juggler applications do not use input devices directly. Instead, proxies are used to provide a generic interface to each device. For each digital and analog input source in each of the devices, the configuration file should include a Digital Proxy or Analog Proxy ConfigChunk.

These two kinds of ConfigChunks are simpler than Position Proxies, since they do not include the transformation properties needed for positional data. The only properties that must be filled in are the name of the device, and the unit number, which is used to distinguish between multiple analog or digital inputs available from a single device.

Many VR Juggler applications (including the test and sample programs packaged with the VR Juggler source and binary distributions) expect there to be a set of digital inputs available with names of the form "VJButton0", "VJButton1", etc., which correspond to the buttons on a handheld wand.

Fewer of the VR Juggler sample applications make use of analog devices, but when they do a similar naming convention (e.g. "VJAnalog0") is used. Of course, nothing about these proxy names is hardcoded - the only requirement is that the config file provide the names that the particular application requests.

As with Position Proxies, Digital and Analog Proxies can be given additional names with the use of Proxy Alias ConfigChunks.

## Add User configuration

We have trackers in our configuration now, but how do we interpret the tracking data? Which sensor is the user's head, and which are his hands?

We create a User ConfigChunk to store a small amount of information about the person using our VR system. When we create displays later on, we'll tell each one which user to draw the view for.

Typically, we will only need one User ConfigChunk. It only has a few properties, as shown in the figure below.

The head position value in the User chunk is simply selected from a drop-down list of all the position proxies (or aliases) we have created.

The eye separation value is used to create the separate left/right views that create a stereo display. It uses the same units as the rest of the VR Juggler coordinate system. The actual physical eye separation varies among individuals, though the value 0.229 (feet) will provide good results for most viewers.

## Add Displays

Now that we have our input devices and users sorted out, we can finally think about the graphics. We need to create a display window for each of the four screens in our example system. We also need to provide some auxiliary information about the graphics pipes and X Windows displays on our system, as outlined below.

## Create a DisplaySystem chunk

There are two folders beneath Displays in the config file editor window. First, we need to create a new chunk under Display System.

VR Juggler's display windows use the idea of pipe numbers - integer values associated with a particular X Windows display - to decide where to open. On a single display system like the average desktop computer, this is simple - the

default X display is labeled ":0.0".<sup>2</sup>

On a multi-pipe system, there are multiple displays, and in our example system the video output of each display is connected to a different monitor. The mapping is shown in the screenshot below (the value "-1" for a display is shorthand for using the value of the `$DISPLAY` environment variable).

In addition to the mapping provided by the `XPipes` property, the Display System also has a `Number of Pipes` property. This property is used for OpenGL-based VR Juggler applications to determine how many drawing threads to create. For `Number of Pipes = n`, VR Juggler will create `n` threads, each servicing one of the first `n` values in `XPipes`.

Extra values in `XPipes` can be useful for defining X displays that are not used for graphics drawing, but which are used for other purposes - e.g. user input windows. See the description on `Configuring simulated devices`, below, for an example.

## Define display windows

The next step - and the last needed to provide minimal functionality - is to create `ConfigChunks` for our display windows. To begin this step, we need some information about how the video is connected to our VR system. For this example there are four screens, each with a stereo display, each powered by a separate X Display on the host computer. We also need to determine where on the screen to position the window so that our graphics show up on the projectors.

With this information on hand, it is time to create the `ConfigChunks` for the display windows. Under `Displays` in the tree view of our configuration file, right-click on `Surface Display` and insert a new chunk. We'll start by editing this chunk and filling in information for the front wall, as shown in the figure. Most of the properties will be the same for each window.

The front wall of our example system is a 12'x9' projection screen positioned six feet in front of our origin. Its projector is connected to the output of the X Display ":0.0" (pipe 0 according to our `Display System` chunk). The X display itself provides a 2304x1024 pixel display area, the rightmost 1024x1024 region of which is sent to the projector. These numbers are all determined by the video settings and physical connections of the computer running the VR system.

In Figure 2.6 we have entered all of this data into the `Surface Display` chunk. The window position and size at the top of the window is easy to understand; the important thing to remember is that the position listed is the position of the window's lower-left-hand corner relative to the display's lower-left-corner. In this case the window is flush with the bottom of the display and shifted over 1284 pixels. We place the window on pipe 0 (":0.0"), and ask for a stereo view with none of the standard window border decorations.

Next we need to provide the information that VR Juggler will use to render graphics to our window. Our display window corresponds to a projection screen in our VR system. The `Corners` property in the display configuration lists the positions of the screen's corners relative to the origin of the VR Juggler coordinate system. You will recall that we decided to use the center of the VR system's floor as our origin. Then, for example, the values for the front screen's upper-right-hand corner are:  $x = +6$  (six feet to the right),  $y = +9$  (nine feet up), and  $z = -6$  (six feet forward, and negative because we use a right-handed coordinate system).

VR Juggler also needs to know where the user is in order to set up the viewing parameters for the display - we set this by selecting our `User` chunk from a drop down list.

We set the `Active` property to true since we want our window to be opened automatically when VR Juggler starts up. We leave the `Tracked` property false. The `Tracked` property is used for configuring displays for head-mounted displays (HMDs) and similar devices - we'll discuss its use later.

There are three more displays to go, but these are easier. Select the `FrontScreen` chunk in the tree view of the configuration file and press the `Duplicate` button. Double-click on the newly created "copy of `FrontScreen`" chunk to edit its values. Rename it to "`LeftScreen`" and change the `Pipe` and `Corners` values appropriately. It can help to remember that the right-side corners of the left screen are the same as the left-side corners of the front screen.

---

<sup>2</sup>The display names are ignored on Windows systems, but the number of pipes is still important.

Repeat the previous step to create configurations for the right screen and the floor.

## Additional features

At this point, our example file has all the information needed to start up a simple, OpenGL-based VR Juggler application. If you've been following along, making a configuration file for your own VR system, you may want to give it a try with one of the sample programs in VR Juggler's samples directory.

The next sections discuss configuration options for using other graphics APIs (primarily OpenGL Performer) and for enabling some of VR Juggler's advanced features, such as dynamic reconfiguration.

## Configuring Performer API features

VR Juggler includes a handful of configuration options specifically for applications that use the OpenGL Performer™ graphics API. To configure these features, open the API-specific folder and create a Performer API chunk. This chunk lets you choose the model files (e.g. in ".flt" format) that should be used to represent the user's head and wand position in simulator mode. Example model files are provided in `$VJ_BASE_DIR/share/Data/models`.

## Configuring audio features

The API-specific folder also contains two ConfigChunk types for configuring VR Juggler's built-in audio capabilities - one for the AudioWorks driver, and one for the SL sound driver. See *Sonix: Juggler Simple Sound Interface* for more information.

## Enabling dynamic reconfiguration

Just below the API-specific folder is the Environment Manager folder. The Environment Manager is part of VR Juggler's dynamic reconfiguration capability. When it is activated, VjControl can connect to the application to monitor its performance or change its configuration.

To enable this feature, first create an Environment Manager chunk. This ConfigChunk has two important properties. The first is the Port - the network port that the application will use to listen for connections. You also need to supply this number to VjControl when it tries to open a connection. This argument is typically an integer between 1024 and 65535. The second important property is Accept Connections. Obviously, this should be set to True if you want to use VjControl's monitoring and control capabilities.

## Logging performance data

Another service provided by the the Environment Manager is performance monitoring of VR Juggler applications. Using VR Juggler's performance monitoring capabilities requires two simple steps.

First, we need to tell the application which kinds of performance data to gather. There are three basic categories:

- `kernel` - collect statistics about VR Juggler's kernel thread. This includes several application-specific functions that are called by the kernel thread.
- `vjGlPipe` - statistics about VR Juggler's drawing threads for OpenGL applications. Includes information about the `draw()` methods supplied by applications.
- `Head Latency` - information about the latency of the user's head position data used to draw the scene.

These categories can be individually activated and deactivated with a Performance Measurements ConfigChunk

(which you can create under the Environment Manager folder in VjControl's config file editing window) - see Figure 2.7.

Secondly, we must tell VR Juggler where to send the performance data once it is collected. This is determined by the Performance Target property in the Environment Manager ConfigChunk. If we are dynamically reconfiguring VR Juggler so that we can view performance while the application is running, the pulldown menu for the Performance Target property will list the network connection between VjControl and the application as a possible target. If that connection is selected, performance data can be viewed in a panel in VjControl (click on the Performance tab).

We can also send performance data to a file for later evaluation. To do this, first create an EM Connection ConfigChunk (found in the Environment Manager folder in VjControl). This chunk is illustrated in Figure 2.8. Here we can select the name of the output file and tell VR Juggler to "activate" it - that is, to open it and write to it. Note that the File Mode property should always be "output" for a log file.

Once we have created the EM Connection for our log file, we can return to the Environment Manager ConfigChunk and select it as our target for performance data. The output file that results from running the application with this configuration can be loaded into VjControl's Performance panel for analysis.

---

# Chapter 3. Configuring other hardware devices

Our step-by-step example above concentrated on a particular hardware setup. In the next several sections, we will discuss configuration for a number of other VR devices.

## Desk or wall-based projection displays

There are a large number of single-display projection systems with a variety of physical configurations - walls, desktops, angled surfaces, etc. All of these display devices can be configured using the same technique discussed above for configuring an SSVR system. Simply create a Surface Display chunk, and fill in the values for the screen/desk's corners relative to the origin of the VR Juggler coordinate system (the origin should typically be a spot on the floor several feet in front of the screen, or possibly directly below a horizontal desk-type display).

## Head-mounted displays

There are a large class of head-mounted displays (HMDs), ranging from helmets to glasses to displays mounted on movable arms. These are also configured for use with VR Juggler with the Surface Display ConfigChunk, but there are two important differences.

The most obvious difference is that these displays do not have a fixed position. Instead, they follow the movements of the user's head. To enable this sort of head tracking, turn on the Tracked property toward the bottom of the Surface Display editor window. Next, select a position proxy for the TrackerProxy property (usually, you'll want to use VJHead for this). Now, the view drawn for this display will change as if it were hooked to a video camera on the user's head.

The other difference with an HMD display is that the values for the Corners property typically won't be the physical position of the displays (typically a few inches in front of the user's face). Instead, the HMD documentation will give display information, such as "108 inch diagonal at 4' distance, 4x3 aspect ratio". This information can be used to calculate reasonable values for the corners, with the origin at the point between the wearer's eyes. (For the above example, the corners values would be  $x = \pm 3.6$  feet,  $y = \pm 2.7$  feet, and  $z = -4.0$  feet).

One other aspect of HMD configuration to mention is that some HMDs use two separate video inputs, one channeled to each eye. If this is the case, each eye should be configured as a separate Surface Display window, with its screen coordinates and so forth. Set the View property to "Left Eye" or "Right Eye" in each, instead of "Stereo". Consult your HMD documentation for more details.

## Using the Trackd™ tracker daemon with VR Juggler

VRCO's Trackd™ is a standalone server (or "daemon") which can manage a large variety of VR hardware devices and supply input data to end-user applications. VR Juggler can communicate with Trackd to provide input data for an application.

Individual devices in Trackd are identified by *shared memory keys* - integer values that are defined in Trackd's own configuration files. Note that configuring Trackd itself is outside the scope of either VR Juggler or this document.

VR Juggler uses two kinds of ConfigChunks for communicating with Trackd:

- TrackdSensor, which is equivalent to a single positional input in VR Juggler. The only information you need to configure a TrackdSensor is its shared memory key (the transformation and other parameters shared by most positional input devices are handled by Trackd's own configuration). Once a sensor is configured, you should at#

tach a Position Proxy to it, as was described for the MotionStar in the tutorial above.

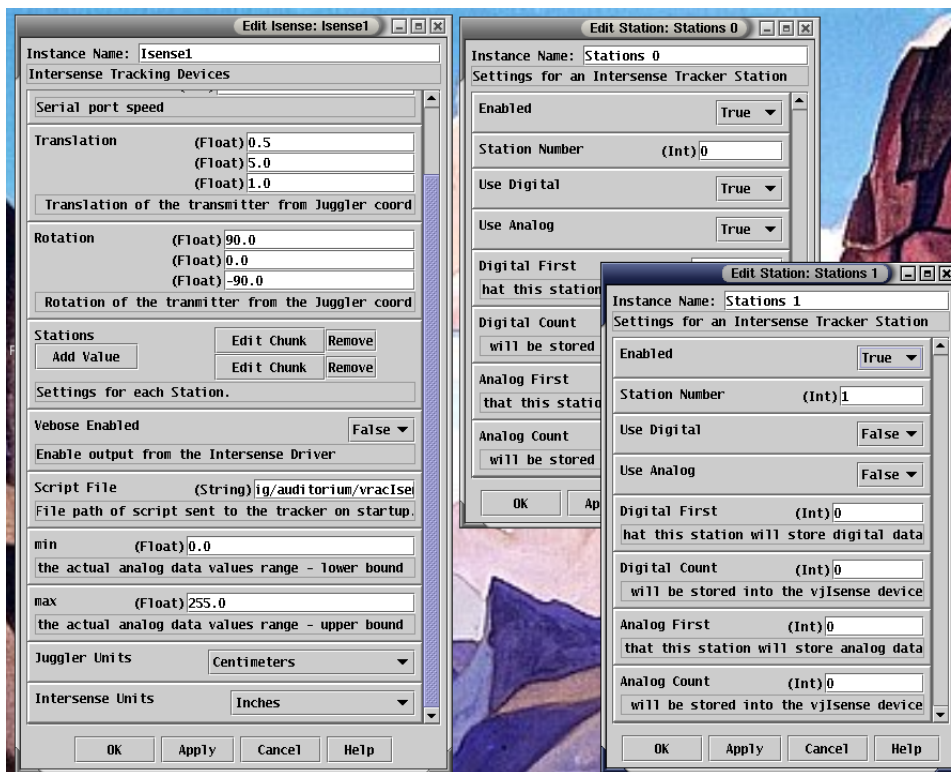
- TrackdController, which presents analog data in a specified range. In addition to the shared memory key, you must supply VR Juggler with the minimum and maximum values the analog controller can return.

A VR Juggler configuration can include multiple sensors and controllers, depending only on the number of devices Trackd itself is configured to use.

## Using InterSense trackers

VR Juggler includes a driver for the InterSense IS-900 wireless tracking system. This tracking system supports multiple "stations", each of which provides a positional input. Some stations also include digital and analog inputs - buttons, sliders, etc.

**Figure 3.2. Configuration for InterSense tracking system.**



The first step in configuring this kind of tracker is to create an ISense ConfigChunk. It has many of the same fields as every other tracker configuration, such as serial port information and translation/rotation values (see the section called “Configure trackers”, above).

The most interesting variation in the ISense driver configuration is the Stations property. This property contains multiple Station ConfigChunks - one for each physical station connected to the InterSense tracking system, as seen in Figure 3.2.

Each station has an integer ID value (which is determined by the order in which it is hooked up to the InterSense system). Since every station provides positional/rotational data, the ISense driver as a whole provides as many unique positional inputs as it has stations. When you attach Position Proxies to the ISense driver, the unit number in

the proxy corresponds to the ID of one of the stations.

The stations may also provide digital and analog data. If the Use Digital property in a station is set to true, VR Juggler's ISense driver will provide its digital data to applications. The Digital First and Digital Count properties determine how a station's digital inputs are mapped to unit numbers that can be used to attach Digital Proxies to the ISense driver. For example, if the Digital First property for a particular station is 3, and the Digital Count is 4, the Digital Proxies with units of 3, 4, 5, and 6 will provide inputs from this station. Values for analog inputs are specified similarly.

VjControl does not check for conflicts between stations. Therefore, you should be careful that you do not specify two stations as providing the same digital or analog data unit.

## Using Fakespace Pinch™ Gloves

VR Juggler includes a driver for the Pinch™ Gloves marketed by Fakespace Labs. Pinch Gloves (available singly or in pairs) have sensors on the fingertips which determine if each finger is touching either the palm of the hand or another finger. The result is a set of binary values, five for each hand.

Configuration of the Pinch Glove driver itself is simple. The glove hardware attaches to the serial port of the host computer, so the ConfigChunk includes serial port and baud rate properties. Most users attach some kind of a tracking sensor to the gloves, so VR Juggler's configuration provides a property for specifying a Position Proxy attached to the glove.

Every VR Juggler input device needs to be attached to a proxy so that it can be accessed by applications. For the Pinch Gloves there are two choices. First, the glove can be treated as a digital input source with five boolean values (10 for a pair). This means that you can attach five Digital Proxies to a Pinch Glove (with unit properties in the range 0 to 5), and use it to emulate a wand.

The other alternative is to attach a Glove Proxy to it. Applications that use the Glove Proxy interface can receive information about the individual finger joint positions of the glove. Of course, because of the capabilities of the Pinch Glove, VR Juggler simply infers these joint positions based on the boolean values for each finger.

---

# Chapter 4. Configuring simulated devices

While the focus of VR Juggler development is on fully immersive VR systems, the reality is that many times such a system is not available. Often, this is because many developers have to share a single HMD or other device. Researchers also frequently need to demonstrate their applications in a wide range of settings, where the usual paraphernalia of VR is unavailable.

For all these reasons, VR Juggler supports a set of simulator objects to replace the usual positional, analog, and digital input devices. By configuring these simulated devices, users can interact with an application using only a keyboard and mouse.

## Creating a keyboard input window

VR Juggler uses keyboard input windows as the source for all simulator data. In order to use simulated devices, a configuration file needs to contain at least one Keyboard ConfigChunk (located with all the other input device ConfigChunks in VjControl's tree display). It may be convenient to use multiple keyboard windows if you are trying to simulate a large number of devices - that way each device can be controlled with the same key bindings, but with a different keyboard window being active.

The *VR Juggler Getting Started Guide* gives an example of running an application with simulated devices and using the keyboard window to control the user's position and wand buttons.

The Keyboard ConfigChunk is relatively simple. Its most important properties are simply the size and position of the window to open. It also has a Display Number property which, like the Pipe property in Surface Displays (described above), refers to one of the displays named in the Display System ConfigChunk. Usually, that property is set to "-1".

The Keyboard chunk has a few more esoteric properties:

- Simulated devices can use mouse movement in addition to keypresses for their controls. The Mouse Sensitivity property controls how mouse movements are translated into keypresses. The value is the number of keypresses that a single pixel of mouse movement should be translated into. For example, 0.1 would mean ten pixels of mouse movement translate into a single keypress. Higher values increase sensitivity. 1.0 is a reasonable initial choice, which can be customized to match the user's tastes.
- The Lock Key property is used to set a key which, when held down, will "lock" the mouse in place in the center of the input window. This feature is useful in environments such as X Windows where moving the mouse out of a window changes the input focus. The Start Locked property is self-explanatory, though its use may seem rather odd. It is generally used when opening a window on a separate display - such as one with a wireless mouse attached, as was described in the section on input devices, above.
- The Sleep Time property controls the behavior of the control thread that manages the keyboard window. It should generally be set to a value between 10 and 50.

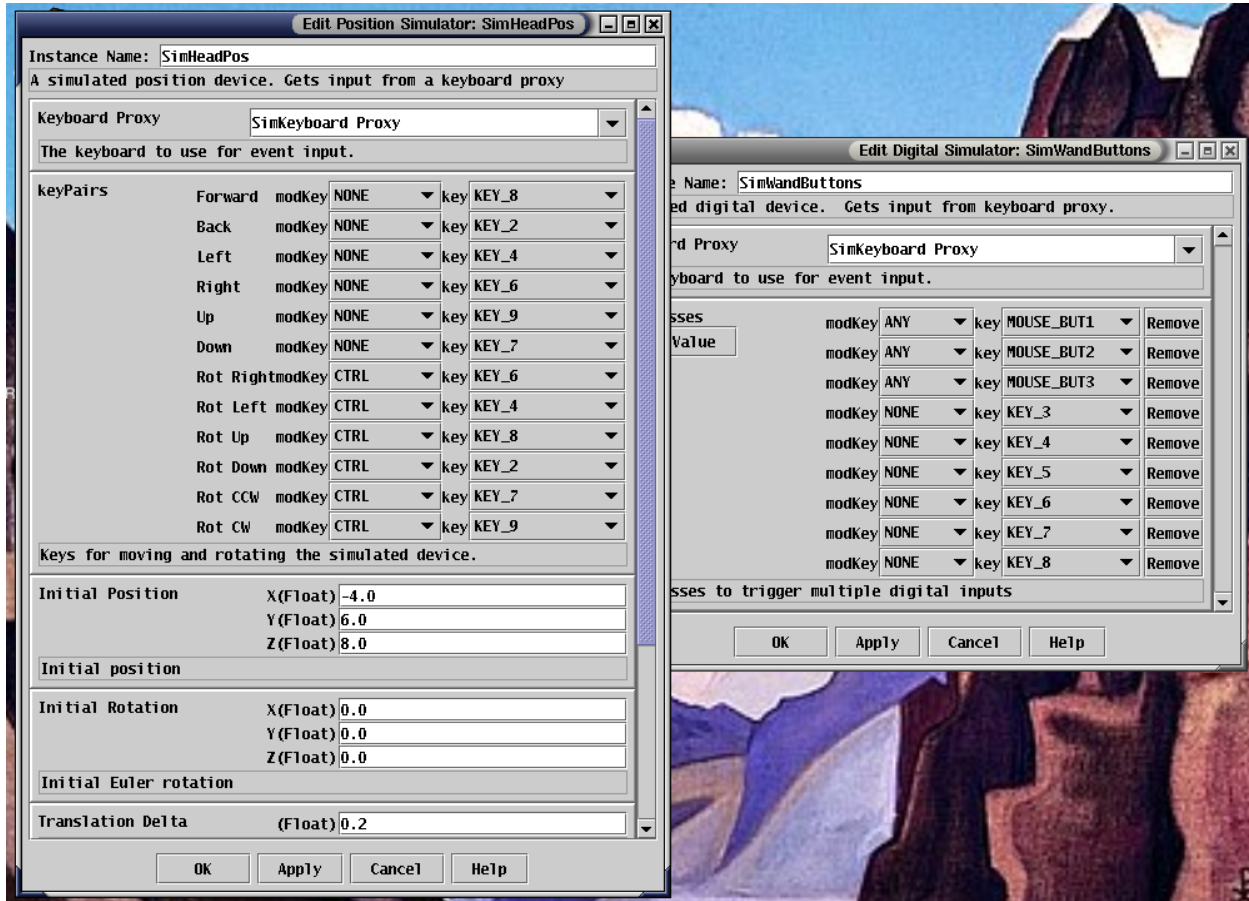
As with every other VR Juggler input device, in order to use a keyboard window you must create a proxy for it. The Keyboard Proxy chunk is very trivial, consisting only of the Keyboard chunk's name.

Note that in VR Juggler 2.0, display windows will also be able to serve double duty as keyboard input sources.

## Creating simulated devices

When editing a configuration file in VjControl, the simulated devices are located in a folder inside the Input folder. There are simulator devices for each kind of input device VR Juggler recognizes.

**Figure 4.1. Configuration of simulated devices. These are the digital and positional simulators used to provide the user's head control and wand buttons in the simstandalone.config file included in the VR Juggler distribution.**



All simulator devices have a few commonalities. Each is configured with the name of a Keyboard Proxy which it uses as a source of user input. Also, each has a set of key bindings. For example, a position simulator will have key strokes assigned for movement in each direction. By comparison, a digital simulator will have a single keystroke to indicate "on", and an analog simulator will have a pair of keystrokes to increase and decrease its value. The ConfigChunks for digital and analog simulators can actually configure multiple instances of each, just by creating additional keystrokes. Some sample simulator ConfigChunks are shown in Figure 4.1. Note how the digital simulator on the right simulates 8 separate digital values, using a combination of mouse buttons and number keys.

Just like every other input device, simulator devices must be connected to proxies before they can be used by VR Juggler. Simulated devices use the exact same proxy ConfigChunks as real hardware devices, and the proxies are configured with the same combination of (simulated) device name and unit number. Note that position simulators (as well as the glove simulator objects) only supply one set of input data, so the Unit property for any proxy referring to them should be 0. Digital and analog simulators, on the other hand, can be configured to support an arbitrary number of input device units.