

# Run-time reconfiguration in VR Juggler

Allen Bierbaum, Carolina Cruz-Neira  
Virtual Reality Applications Center  
Iowa State University  
{allenb, carolina}@vrac.iastate.edu  
<http://www.vrjuggler.org>

## Abstract

Most current virtual reality system designs use an overly static method of configuration. The configuration is specified in advance and once an application has started running in the system, it is not possible to change the its initial setup. This limits the abilities of the VR system to adapt to runtime changes and recover from system failures.

The VR Juggler virtual reality development environment has been designed from its inception to be dynamic at run-time. VR Juggler allows components to be configured and replaced at run-time while allowing applications to execute transparently.

This paper presents the motivation for the need for run-time reconfiguration and describes VR Juggler's implementation of such a system.

## 1. Introduction

Most current virtual reality (VR) systems do not allow users to make run-time changes to modify initial settings. A user configures the system before running an application, and the configuration remains static for the duration of the application. For example, the initial settings specify how many projection surfaces to display and what type of tracking system to use. Each facet of the system is specific a priori. There is no way to modify these settings once the software system has started.

This is because, most VR control software relies upon having all configuration information when the application starts. If the user needs to change a system setting, they have to shutdown the running application, change the configuration parameters, and then restart the application with the new parameters. They may have to repeat these steps many times to get the system into the correct configuration.

Requiring static configurations limit the ability of a VR system to adapt and change to new requirements.

This paper presents the motivation for the need for run-time reconfiguration and VR Juggler's implementation of such a system. We start by describing a generic VR

system and the need for run-time capabilities to modify the initial settings. Then we cover previous efforts within this area and present our design approach within the VR Juggler framework.

### 1.1 General VR system components

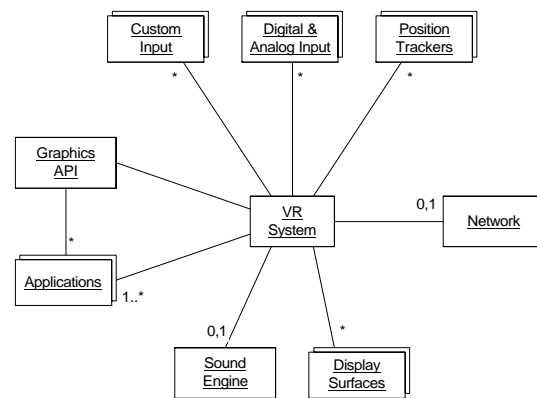


Figure 1: VR system components

For the purposes of this discussion, a VR system is defined as the combination of the hardware and software that enables developers to create and execute VR applications. As shown in Figure 1, a VR system has many separate components that work together. A VR system receives input from many devices, some of which may be custom to a specific application or user. A VR system may have one more running applications in the system, and each of these applications interacts with a corresponding graphics API. The application(s) provide multi-sensory output to the user; in the case of this model, output is represented by the display surface for visual output and the sound engine for auditory output. In addition to these components, the VR system may be connected to a network to allow for remote communication.

As can be seen, current VR systems contain much inherent complexity. Complex systems such as this need to allow for easy adaptation to new requirements, changed settings, refined user preferences, and the ability to

recover from failures. Because of this, we feel run-time reconfiguration is a necessity in a VR system.

## 1.2 Uses of run-time reconfiguration

### 1.2.1 Setup

Run-time reconfiguration can prove invaluable when setting up and configuring a new system. Reconfiguration allows users to change device configuration parameters while an application is running. For example, users can use this ability to interactively tweak tracker settings at run-time. A person setting up a system can run a calibration program that draws a coordinate axes at the position of a tracker. By using this visual feedback, a user can interactively test whether the offset and rotation parameters for the current tracking system have been configured correctly. This type of interactive testing is invaluable in determining correct system settings.

Reconfiguration also permits display setting to change at run-time. This allows users to configure new projection surfaces and display parameters while running test applications.

First, the user specifies an initial system configuration that may include information such as the tracking system used and any other information that they believe to be correct. This initial configuration also includes speculated settings for the display surfaces in the environment. The user then starts a test application in order to try the settings. Once the application is running, the user can then interactively change the settings of the active display surfaces, and even add new displays or remove current ones. They can change parameters such as the size, location, and a variety of other projection parameters, all while observing the result of the changes.

Reconfigurable VR software systems also enable the use of advanced projection systems where the physical settings of the projection surface itself are changing at run-time. Examples of this type of system include desk-based systems with a movable projection surface and large scale CAVE-like devices that allow users to move the walls. Run-time reconfiguration allows users to reconfigure the desk projection surface while an application is running; or if a driver is available, the running system can actively monitor the desk's current settings and automatically update the projection parameters to reflect any changes.

### 1.2.2 Software and hardware testing

A reconfiguration VR system can also be very helpful when testing new software and hardware.

Reconfiguration reduces the turn around time for testing multiple configurations while developing

applications. Many times VR applications take a lengthy amount of time to load due either to loading large models or the time required to start input devices. When debugging an application, there is no reason to bring the application up in a full VR environment every time the program starts. Instead, a developer can start the application in a simulated environment to quickly test the program. If the application is working in the simulated environment, then instead of restarting it with new configuration information, run-time reconfiguration allows a user to simply change the configuration so that it is running in the full VR system.

Developers can make use of this flexibility to test an application using many different VR system configurations. For example, it may be helpful to start the application first in a simulator, reconfigure it so that it is running with an HMD, and then reconfigure it again so that it is using a CAVE or some other large-scale projection environment. Applications can be tested in all of these environments without ever halting execution.

### 1.2.3 Robustness

Run-time reconfiguration allows for recovery from failure of software and hardware components of the VR system. Each components of a reconfigurable system accesses the other components through proxies that shield the system components from each other. This separation of components is what allows the system to handle the addition, removal, replacement, or even the failure of any software component that makes up the system. For example, if a display is closed, this happens transparently to any running applications because the applications do not have direct references to the displays.

The design of run-time reconfiguration also allows for robust testing of hardware because the system is able to deal with device failures. The design of a reconfigurable system necessitates extreme separation of the application from the actual hardware being used. Because of this, the application will not crash if a hardware input device fails. Instead the application will continue to run, and will simply wait for the device to begin working correctly again. For example, an experimental hardware device could be restarted or even re-wired as many times as necessary without interrupting application execution.

### 1.2.4 Performance tuning

Run-time changes can be used to tune performance of VR applications. By analyzing the performance of a running application, and adjusting the current system configuration based upon these measurements, users can change a VR system configuration to achieve better performance. For example, device drivers could be moved to lightly loaded systems or have parameters

changed in such a way that the driver requires less system resources. As another example, consider advance graphics architectures where it is possible to change the parameters of the graphics hardware. Users of run-time reconfiguration can exploit these abilities in order to find more optimal settings for the VR system's graphics hardware.

VR Juggler provides a performance monitoring system, that when used in combination run-time reconfiguration can achieve exactly this type of performance tuning [1].

### 1.2.5 Application adaptations

Reconfiguration is not limited to only the VR system. VR applications can also take advantage of the abilities afforded by such a system. Since the VR system provides the infrastructure, it becomes much easier to write applications that are reconfigurable as well. Applications can allow certain application parameters to be configured using the reconfiguration system.

VR system reconfiguration can be used to switch applications at run-time. This means that only the application changes, but the rest of the VR system remains the same. When an application becomes active in the system, it can have associated configuration information that changes the system to the setup that it expects. For example, the original application may have used a tracker and several digital inputs for interaction, but the second application requires a glove for input. This second application can give the system a corresponding configuration request to reconfigure the system to suit its needs.

In some cases, it is also possible to run multiple applications simultaneously where they all share the same VR system and the same configuration information.

Runtime reconfiguration can also be used to change application specific parameters such as models loaded or interaction methods used. For example, an application can be written where the user can remotely change the model that is being viewed in the VR environment and the navigation method that is being used. Because the run-time reconfiguration system is being used to make these application changes, they can come from any entity be it a remote controlling interface, another application that is running, or the local application it self.

In a VR system that allows for multiple simultaneous users in a single environment, run-time reconfiguration allows user to exchange control. An example would be an application that was not written with multiple users in mind. As such, the application would only expect to be controlled by one input device. Using run-time

reconfiguration, the two users can exchange which user's interaction device is active in the system.

This same idea can be used to enable multiple tracked users to use an environment that only has support for a single tracked user at one time. In such an environment, the run-time reconfiguration system can be used to choose which tracked user is "active" and thus controlling the environment at any given time.

## 1.3 Previous work

Previous work in VR system reconfigurability has centered on separating input devices from the VR system. An example of this the trackd API [2] that is used to provide a common method for acquiring data from many types of input devices. The trackd API specifies a format to write tracker information into shared memory in a way that applications can easily read. This abstraction allows users to change the actual input hardware being used. TrackD is used in several of today's most widely used VR software systems [3][4][5].

This type of separation does help to shield applications from input device issues, but does not provide the ability to change other components of the VR system.

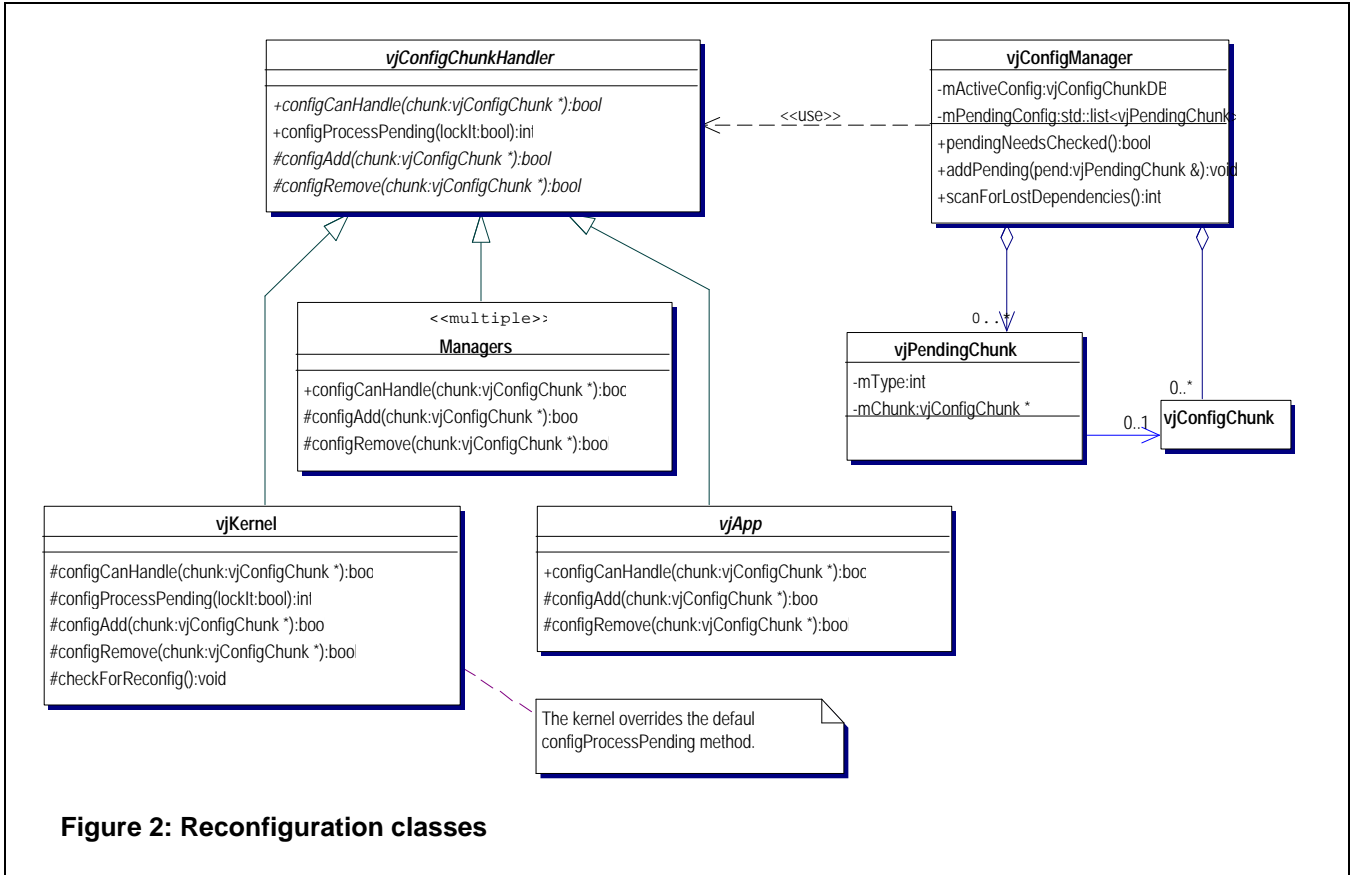
## 2. Run-time reconfiguration in VR Juggler

VR Juggler is an open-source virtual platform for the development of VR applications [6][7][8]. VR Juggler has been under active development at the Virtual Reality Application Center at Iowa State University for the past 3 years.

Since its inception, VR Juggler has been designed to facilitate run-time reconfigurability. VR Juggler is based on a micro-kernel [9] architecture of loosely coupled objects. The system implements all its services in *manager* objects that are controlled by the *kernel*. There are managers to handle input devices, display settings, configuration information, graphics API interfaces, and to communicate with the external applications. The kernel maintains a reference to all of the managers in the system and controls the interactions between them. This design allows each element in the system to be inserted, removed, or reconfigured at run-time thus allowing the virtual platform to alter its behavior at run-time.

The next sections describe how run-time reconfiguration is implemented in VR Juggler.

### 2.1 Pending configuration queue



In VR Juggler, a single unit of configuration information is contained in an object called a *config chunk*. Config chunks are passed to the VR Juggler kernel from the user application and from a java-based control program called *vjControl*

Reconfiguration is implemented using a pending config queue that is contained in the config manager (see *vjConfigManager* in Figure 2). As with all other managers, the kernel controls this manager. The system initially starts with only the kernel and several system managers instantiated. When the kernel receives a new config chunk, it passes it to the config manager which adds the config chunk to a pending reconfiguration queue.

Each frame, managers and any other interested component in the system check the reconfiguration queue for new entries. If there are new entries, and the querying component knows how to use the new entry, then the component processes the entry. It is the responsibility of the configuring component to alert the kernel to the change in configuration.

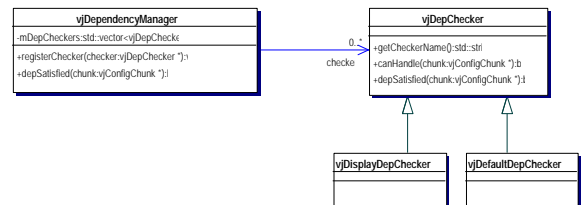
This method of processing the configuration information is based on the chain of responsibility pattern [10]. It differs only in that the handlers in VR Juggler do not have a successor chain. Instead, the kernel keeps a list of all possible handlers in the system. The system then traverses this list to find a valid handler for a given

reconfiguration config chunk. It is also possible to directly query the config manager if an object would rather not have the kernel manage its reconfiguration.

## 2.2 Class interfaces

VR Juggler provides a common framework for processing configuration entries in the config queue. A class can make use of this framework by inheriting from the *vjConfigChunkHandler* class (see Figure 2). Once the class has these abilities, it is only necessary to provide custom implementations for the functions in the framework.

## 2.3 Advances Abilities



**Figure 3: Dependency checking classes**

As implemented in VR Juggler, the reconfiguration system has several advantages beyond basic reconfiguration.

VR Juggler allows for smart dependency checking between system components. Dependency checking refers to the process the system goes through when it runs a check to see if all the system resources required by a new configuration request are available before it allows that request to be processed. These checks are implemented in the `vjDependencyManager` using customized dependency checker classes derived from the `vjDepChecker` class (see Figure 2). Developers can make these dependency-checking classes as simple or complex as necessary for a given dependency test.

VR Juggler also allows for smart unloading of system components. This is needed because in a reconfigurable system it is possible to remove a component that another component relies upon to function correctly. Smart unloading allows dependent components to reconfigure themselves dynamically to modify their dependency if possible. If the dependencies cannot be satisfied by modification, then the component is unloaded and placed back into the pending queue until the dependencies are satisfied.

### 3. Conclusions

Results of the initial implementation have been very positive. We have been able to create a very flexible architecture that allows for the majority of system components to be reconfigured at run-time. We are still refactoring the design to increase flexibility and add more reconfiguration abilities to the system.

One area of difficulty that we have found is that many other application programming interfaces (APIs) used in a VR application do not lend themselves well to reconfiguration. Some graphics APIs (such as Iris Performer) expect the be in complete control of system resources and as such do not provide a way to release the API's resources when VR Juggler does not need them any more. We are actively pursuing ways to alleviate these problems by communicating with API developers and by investigating systems that are more flexible.

### 4. Future work

We are currently investigating ways to use componentization to make the VR Juggler system truly component based. Once this is completed, we should be able to reconfigure the few sections of the system that are not currently reconfigurable.

The current implementation has a few difficulties unloading complex dependencies. We are working on ways in which to make this work more completely.

We are also interested in further exploring application reconfigurability and ways to make it easier to write reconfigurable application.

### 5. References

- [1] C. Just, C. Cruz-Neira and A. Baker, "Performance Measurement Capabilities of VR Juggler: Real-Time Monitoring of Immersive Applications," *Fourth International Immersive Projection Technology Workshop*, Ames, IA
- [2] "TrackD API," <http://www.vrco.com/products/trackd.html>
- [3] C. Cruz-Neira, T. DeFanti and D. Sandin, "Surround-Screen Projections-Based Virtual Reality: The Design and Implementation of the CAVE," *ACM SIGGRAPH*, pp. 135-142
- [4] "WorldTookKit Release 8: Technical Overview," [www.sense8.com](http://www.sense8.com)
- [5] "Division homepage," <http://www.division.com>
- [6] "VR Juggler Homepage," <http://www.vrjuggler.org>
- [7] A. Bierbaum, "VR Juggler: A Virtual Platform for Virtual Reality Application Development," Master's thesis, Iowa State University, 2000.
- [8] A. Bierbaum and C. Just, "Applied Virtual Reality Course Notes: Software Tools for Application Development," *SIGGRAPH 98*, pp. 3-37-3-43
- [9] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal, "Pattern-Oriented Software Architecture: A System of Patterns," 1996
- [10] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns," 1995