Control Software for the productive Use of distributed IPT Installations

H. Scharm*, F. Maurer**, D. Neuber, R. Löffler, D. Rantzau, D. Banek

SGI Professional Services, Practice Immersive Solutions, Germany
[*holgers@sgi.com, **fmaurer@sgi.com]

Abstract. Over the last few years the number of industrial immersive projection technology (IPT) installations has grown rapidly. Most of these systems are powered by one or more SGI Onyx graphics computers with several independent graphics pipes and quite a few CPUs. Since these machines represents a big investment, many installations are designed to make use of the Onyx graphics power at different presentation areas and meeting rooms. In this paper we will introduce a new client/server approach for controlling such distributed IPT installations where multiple presentations and work sessions run simultaneously and where different users access the resources of the facility in parallel. We first discuss the necessity of multiple configurations of the IPT hardware for different kinds of presentations and applications. After this, we explain the special requirements for the control of distributed multi-user Reality Center™s ("Reality Center" is a trademark of SGI). We then present the client/server architecture of our new Reality Center management software (RCMS). Finally, we will give an outlook for possible future extensions of the software to meet individual requirements.

1 Introduction

Over the past few years the automotive industry as well as the oil and gas industry have adopted immersive projection technology (IPT) in productive scenarios. Virtual Reality (VR) installations are used as a communication tool and work place for interdisciplinary engineering groups working on different tasks. The benefits of VR technology combined with IPT are beginning to be visible throughout the engineering process. Multi-channel projection systems known as Powerwalls and CAVE™s are setting the new standard for collaborative, interdisciplinary visualization in virtual product development processes. It is interesting to note that especially in productive industry sites, the number of the IPT systems implemented is growing. Today it is common, that several autonomous IPT installations are placed in one VR service center as a sharable resource for the whole site and sometimes also the companies suppliers. A good example for this trend is the VR Center of the DaimlerChrysler AG, MTC in Stuttgart, Germany [1]. It features five independent IPT systems driven by a single large SGI Onyx 2 image generator (currently 14 pipes, 60 CPUs) and quite a number of media devices. The usage of such centers is typically not only limited to VR related work, it is merely used as an universal media center, where slide presentations, video or DVD playback, video conferencing and different – not only VR - software applications co-exist at the same time. Engineers use these facilities on a daily basis in parallel, while utilizing the available media resources in an intensive manner.

The problem that arises with the growing size of centers is the necessity to control the complex set of interconnected devices and media systems. Especially in a service oriented work model it is an absolute requirement that even users without detailed knowledge of the system are able to switch the installation to the operation modes they need, without depending on trained VR center staff. By providing a flexible control software as a part of the SGI Professional Services in Reality Center [2] projects, we believe that we can strongly contribute to the acceptance of immersive projection technology in places with high productivity needs.

2 State of the Art: Media Control Applications

Media center control is not a new kind of application. Today, there are quite a number of commercial and research solutions available, although they do have selective objectives. One of the more popular commercial solutions is to use dedicated control hardware and touch panels to operate media centers ([3] as a representative for this kind). One disadvantage of this technology is the need for extra, proprietary hardware, and therefore additional expenses. Additionally, these solutions generally offer no management of concurrent user access and no control support for more sophisticated hardware, like an SGI Onyx2 or Onyx3000.

The majority of installed center control solutions is based on client/server web technology, where a web server acts as the central control instance. On the client side any web browser can be used to access the system. The actual device driver controls are encapsulated in CGI web server scripts and the center logic is mapped on HTML (see figure 1).

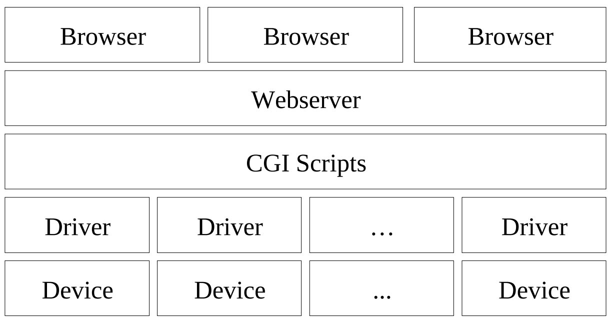
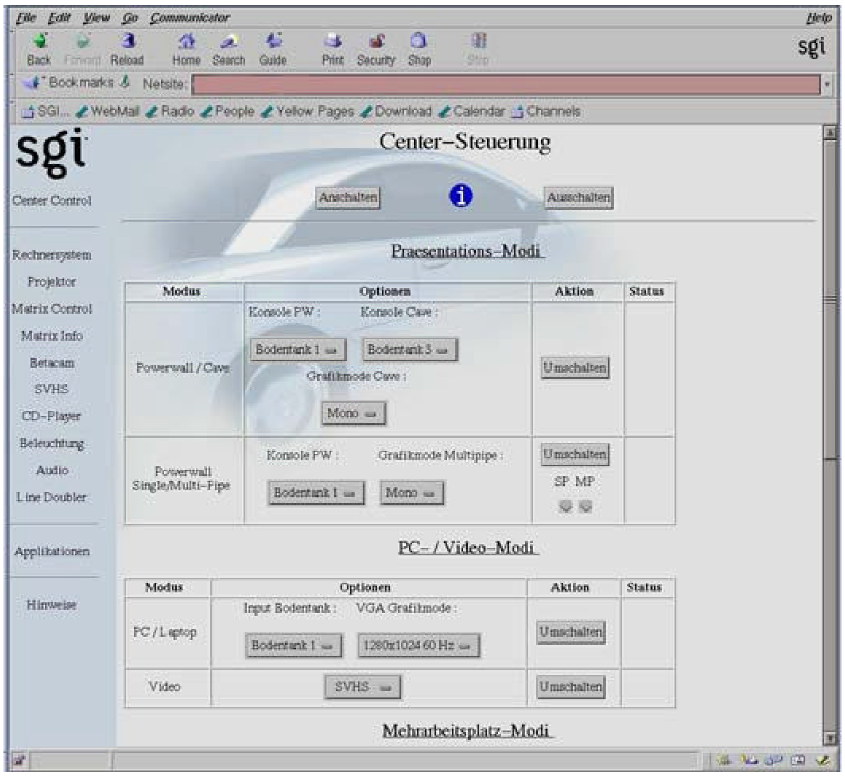


Fig. 1. Typical HTML/CGI control software architecture

Normally it is also possible to create macros (for example with a scripting language like PERL) to setup the hardware configuration for a particular application. These modes can be invoked smartly by a single button click in a web browser. Some of the web pages also use Cookies and/or Javascript code.

In the past, SGI Professional Services has delivered this kind of software based on HTML, CGI and PERL scripts along with most of the Reality Center projects. Figure 2 illustrates the web interface of the center control for an installation in the automotive area. On the left side a navigation bar, with a list of control possibilities can be seen. The entry point for the user is a center control web page (as currently visible) for the most common tasks, additional pages are available for device control and individual tasks. Users can choose the operation mode they need and with some optional parameters, they can switch to the new mode just by one single button click.



Center–Steuerung

Praesentations–Modi

Modus	Optionen	Aktion	Status
Powerwall / Cave	Konsole PW : Bodentank 1; Konsole Cave : Bodentank 3; Grafikmode Cave : Mono	Umschalten	
Powerwall Single/Multi-Pipe	Konsole PW : Bodentank 1; Grafikmode Multipipe : Mono	Umschalten SP MP	

PC– / Video–Modi

Modus	Optionen	Aktion	Status
PC / Laptop	Input Bodentank : Bodentank 1; VGA Grafikmode : 1280x1024 60 Hz	Umschalten	
Video	SVHS	Umschalten	

Mehrarbeitsplatz–Modi

Fig. 2. Example of a web based center control software

Nevertheless, the software as presented before was not able to grow in the same way as the center installation. Due to technical limitations, the HTML based approach is not able to fulfill the new requirements and demands. One major disadvantage is the lacking support for proper status updates to indicate system changes inside the web page (as push technology is required therefore). Another drawback is the enormous maintenance effort necessary, if the underlying HTML/PERL code base has to be expanded and adapted to the growing center environment needs.

3 Requirements for distributed IPT Installations

The main challenge with controlling a distributed IPT installation is that different users will try to make use of the same hardware resources at the same time. Some of the available resources are actually sharable such as a video switching matrix with multiple input sources and output streams, others are atomic per se such as a simple video recorder device. Accessing resources without a dedicated resource locking mechanism causes frustration since work sessions and presentations could be interrupted unintentionally. Another problem is the protection of confidential data. This critical content could be disclosed by broadcasting the video signals of a presentation to another projection system. Last but not least a differentiated access to IPT resources can hardly be implemented using a HTML client.

Changing demands, requirements and the technological limitation of the available approaches motivated us to develop a new kind of control software. The functional requirements can be summarized as follows:

- Support of all controllable devices, including sophisticated ones, like the Onyx image generators.
- Resource locking mechanism to avoid a sharing violation on media resources.
- Independence of proprietary hardware. ‘Make best use of the hardware already available’.
- Dedicated functionality for device and system maintenance operations.
- Multi-user capabilities and user management.
- A tailored user interface, taking into account different skill levels.
- Session management. Users must be able to get an overview about the current workload of the center and administrators should be able to manage the sessions.
- Platform independence for the client implementation.
- A mechanism to easily reconfigure and extend the system configuration.

4 The RCMS Approach

As a consequence from above, we introduce the following client/server architecture as a new kind of control software for IPT installations (see figure 3). The implementation based on this architecture uses the internal working title ‘Reality Center Management System’ (RCMS).

The RCMS server consists of the modules:

- User management (authorization handling),
- Resource management (resource locking and releasing),
- Device management (device control plug-ins) and
- System configuration (device and mode description in XML)

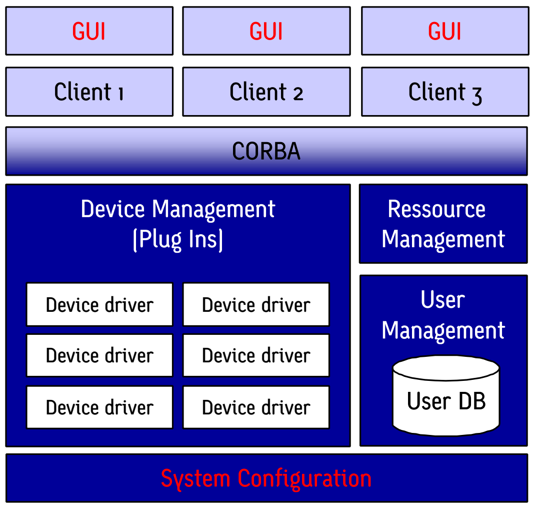


Fig. 3. The RCMS architecture

Both client and server are implemented with JAVA2 technology [4]. For the communication between client and server a CORBA layer is used. This has led to the following features:

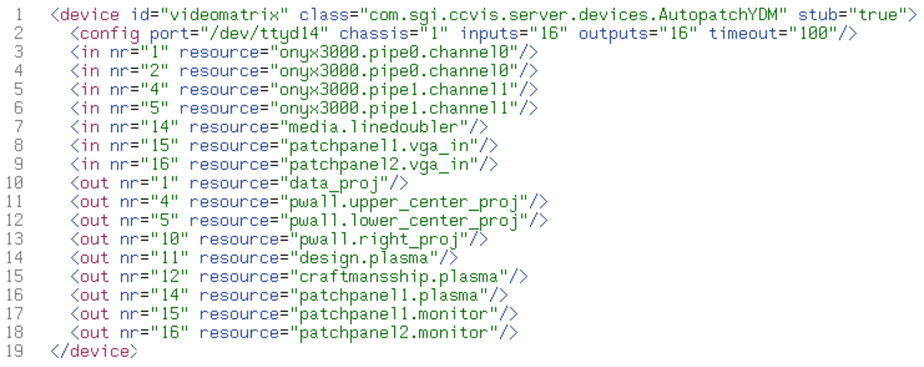
- Clients can run on different platforms (Windows, Linux, IRIX and others).
- The server can run on different hardware platforms.
- No additional proprietary hardware is needed (provided that all controllable devices have a serial connection to the server platform or to a standard terminal server).
- Users cannot disrupt other sessions since all used resources are locked.
- A login procedure for all users secures the access to the system.
- The client GUI behavior can be customized using XML.
- National languages are fully supported (standard properties).
- A hardware upgrade does not result in programming effort. The reconfiguration can be done using XML.

To reflect the diversity of IPT installations, a plug-in concept is used for the user interface as well as for the interface to the hardware, which has to be controlled. This approach offers a maximum in flexibility and extensibility, while leaving the core system untouched if the software is adapted to new installations. The setup of the software is done via XML files. These files store the following information:

- The configuration of every single device (Fig. 4)
- The configuration of the client user interface
- The different application modes of the installation (Fig. 5)
- A hierarchical set of resources

The hierarchical structure of resources is used to be able to lock single devices like a projector as well as whole parts of an IPT installation like a Powerwall.

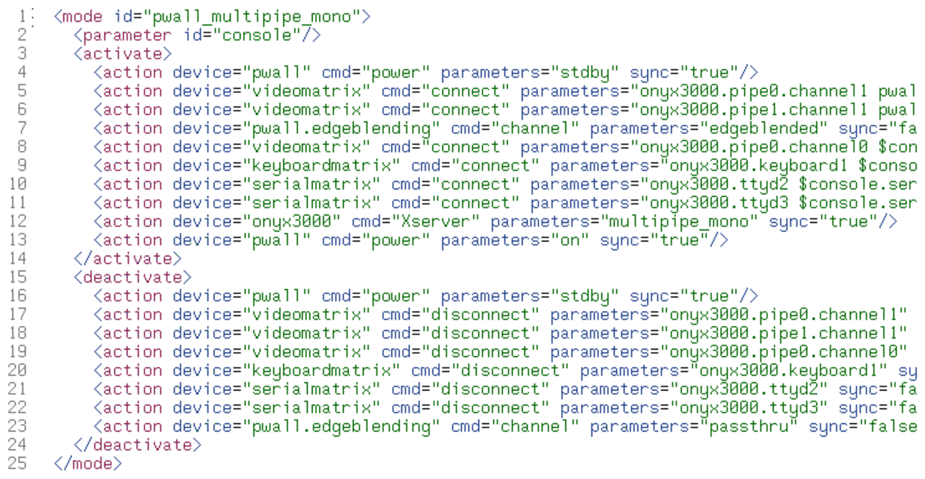
The configuration of the devices is described using an XML scheme. An example of a single device configuration is illustrated in figure 4. By using the reflection API of the Java programming language, it is possible to extend the system setup with new or modified device drivers without affecting the application core system. Therefore it is necessary to incorporate the Java device driver class as an attribute of the XML device element. The devices used in center installations are completely different, starting from simple ones up to very sophisticated ‘devices’ like an Onyx. Therefore it is not possible to define a generic XML device scheme, that fits for all devices. For that reason the parsing of the XML device file is performed by a generic parser. Each time a new device element is parsed, a corresponding device XML parser class is loaded at runtime and the concrete parsing task is passed to it. The attribute “stub” of the device element is a feature to test the configuration of the devices without having physical devices attached to the system. This is a very useful feature to test the complete system configuration before deploying it at a customer’s site. The “config” element contains some attributes and physical settings of the device, like the number of input and output ports of the video media switch. The “in” and “out” elements that follow, establish the mapping between the physical cabling of the media matrix and the input and output ports of the matrix switch.



```xml
<device id="videomatrix" class="com.sgi.ccvis.server.devices.AutopatchYDM" stub="true">
  <config port="/dev/ttyd14" chassis="1" inputs="16" outputs="16" timeout="100"/>
  <in nr="1" resource="onyx3000.pipe0.channel0"/>
  <in nr="2" resource="onyx3000.pipe0.channel0"/>
  <in nr="4" resource="onyx3000.pipe1.channel1"/>
  <in nr="5" resource="onyx3000.pipe1.channel1"/>
  <in nr="14" resource="media.linedoubler"/>
  <in nr="15" resource="patchpanel1.vga_in"/>
  <in nr="16" resource="patchpanel2.vga_in"/>
  <out nr="1" resource="data_proj"/>
  <out nr="4" resource="pwall.upper_center_proj"/>
  <out nr="5" resource="pwall.lower_center_proj"/>
  <out nr="10" resource="pwall.right_proj"/>
  <out nr="11" resource="design.plasma"/>
  <out nr="12" resource="craftmansship.plasma"/>
  <out nr="14" resource="patchpanel1.plasma"/>
  <out nr="15" resource="patchpanel1.monitor"/>
  <out nr="16" resource="patchpanel2.monitor"/>
</device>
```

Fig. 4. Example device (video media switch) in devices.xml

Figure 5 illustrates an example of a XML mode description. The multi-pipe mode on the Powerwall will be identified further on by an unique identifier string “id”. After that, the parameter element of the mode indicates that a customizable parameter is required from the client. In this case, the location of the console where the user wants to connect the control terminal is necessary. Depending on the user role, the client implementation either uses a default value for the actual console location or asks the user to choose from a list of possible locations during a mode switch request. For the following mode description, the string “console” is substituted with the user value. The XML description is now followed by two main sections: one section with a set of actions for the activation of the mode (constructor), another section with a set of actions for the proper deactivation (destructor). Each action line represents a device command, for example the videomatrix (Line 5 and 6) to connect the video output signal of the Onyx to the input ports of the soft edge blending box. The sync attribute of the action element is used to control the multi-threaded execution of the device actions. Some of the actions are not critical and can be run in parallel to speed up the total mode switch time. Other operations require some other actions to be completed and the execution of the sync command waits until all previous actions have been performed.



```
 1  <mode id="pwall_multipipe_mono">
 2    <parameter id="console"/>
 3    <activate>
 4      <action device="pwall" cmd="power" parameters="stdby" sync="true"/>
 5      <action device="videomatrix" cmd="connect" parameters="onyx3000.pipe0.channel1 pwal
 6      <action device="videomatrix" cmd="connect" parameters="onyx3000.pipe1.channel1 pwal
 7      <action device="pwall.edgeblending" cmd="channel" parameters="edgeblended" sync="fa
 8      <action device="videomatrix" cmd="connect" parameters="onyx3000.pipe0.channel0 $con
 9      <action device="keyboardmatrix" cmd="connect" parameters="onyx3000.keyboard1 $conso
10      <action device="serialmatrix" cmd="connect" parameters="onyx3000.ttyd2 $console.ser
11      <action device="serialmatrix" cmd="connect" parameters="onyx3000.ttyd3 $console.ser
12      <action device="onyx3000" cmd="Xserver" parameters="multipipe_mono" sync="true"/>
13      <action device="pwall" cmd="power" parameters="on" sync="true"/>
14    </activate>
15    <deactivate>
16      <action device="pwall" cmd="power" parameters="stdby" sync="true"/>
17      <action device="videomatrix" cmd="disconnect" parameters="onyx3000.pipe0.channel1"
18      <action device="videomatrix" cmd="disconnect" parameters="onyx3000.pipe1.channel1"
19      <action device="videomatrix" cmd="disconnect" parameters="onyx3000.pipe0.channel0"
20      <action device="keyboardmatrix" cmd="disconnect" parameters="onyx3000.keyboard1" sy
21      <action device="serialmatrix" cmd="disconnect" parameters="onyx3000.ttyd2" sync="fa
22      <action device="serialmatrix" cmd="disconnect" parameters="onyx3000.ttyd3" sync="fa
23      <action device="pwall.edgeblending" cmd="channel" parameters="passthru" sync="false
24    </deactivate>
25  </mode>
```

Fig. 5. Example of a system mode in modes.xml

5 Sample Installation

Figure 6 illustrates a screenshot of a working draft for a sample client IPT installation with a three channel Powerwall. The main interaction paradigm of this client GUI is drag-and-drop. The application icons on the left can be dragged onto the visual representation of the Powerwall in the client’s center area. If all needed resources are available, the application mode will be switched and a corresponding application image appears on the Powerwall. Otherwise the user gets information about the reason for the failure. This information is tailored to the current user role (i.e. knowledge and authorization level). Remote parts of the IPT installation are located in the right column. In this case, there are additional CAD workplaces in a nearby office where designers and stylists are able to use free graphic pipe resources of the Onyx to connect their CAD terminal to. In the lower area, single devices can be controlled using a set of controls. The availability of controls also depends on the current user role. If for example a user has no permission to change the settings of a device, the according control will not be visible for him.

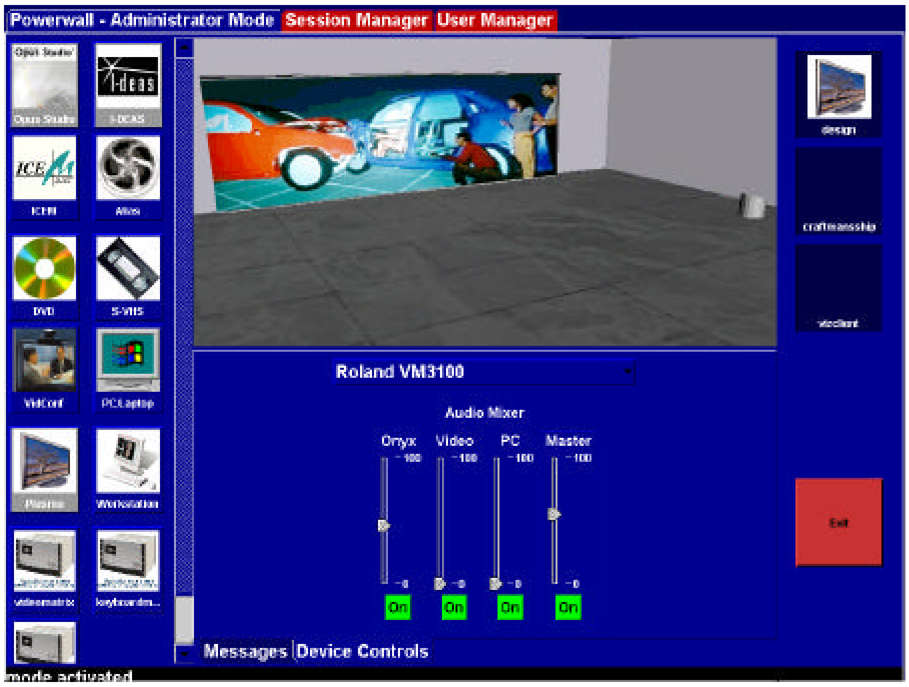


Fig. 6. Sample client GUI

A first commercial version of the software will be available in spring 2001. Future work can include dedicated functionality to support the daily work tasks of center operators. Examples are connectivity features of the center software to a calendar tool or an accounting system to keep track of the actual center usage times. There are some interesting Java Specification Requests (JSR 059) [5] proposed for the upcoming Java Releases. API extensions like XML Parsing (005), Preferences API (010), Logging API (047) and Network Launching API (056) are very interesting for the standardization of the RCMS software implementation. Some of the APIs can replace our own implementation (like logging) or external software modules (like Xerces-J [6] for XML parsing), others can extend the functionality (like the Java Network Launching Protocol).

6 Conclusion

In this paper we described a new kind of control software for IPT installations. It is in particular useful for distributed installations and enables engineers to focus on their work with the applications and tools rather than on the configuration and resource sharing of the hardware installation. The design of the software reflects the lessons we have learned while installing and operating Reality Centers for our customers. We are sure that this software approach is a step towards more user friendly and hence productive industrial IPT installations.

Acknowledgements

We'd like to thank all our partners and colleagues at SGI, especially the RCMS development team, for their enthusiasm and effort bringing the software to life. Finally, many thanks to all the reviewers for their valuable input.

References

1. DaimlerChrysler Virtual Reality Center (VRC).
   http://www.daimlerchrysler.com/index_e.htm?/specials/virtual/virtual5_e.htm
2. SGI Reality Center™ Advanced Visualization Facilities.
   http://www.sgi.com/realitycenter
3. Crestron Remote Control Systems.
   http://www.crestron.com
4. Sun Java Technology.
   http://www.javasoft.com
5. Java Specification Request (JSR).
   http://java.sun.com/aboutJava/communityprocess/jsr
6. Apache XML project.
   http://xml.apache.org