A REAL-TIME AUDIO RENDERING SYSTEM FOR THE INTERNET (iARS), EMBEDDED IN AN ELECTRONIC MUSIC LIBRARY (IAEM)

Christopher Frauenberger

Institute of Electronic Music and Acoustics
University of Music and Dramatic Arts Graz,
Inffeldgasse 10/3, 8010 Graz, Austria
frauenberger@iem.at

Winfried Ritsch

Institute of Electronic Music and Acoustics
University of Music and Dramatic Arts Graz,
Inffeldgasse 10/3, 8010 Graz, Austria
ritsch@iem.at

ABSTRACT

The internet Audio Rendering System (iARS) is an Internet browser extension extending the browser's capabilities with real-time signal processing. The proposed system allows to receive audio streams from the Internet and apply various audio algorithms with no additional computational power needed from the server. iARS is part of the Internet Archive of Electronic Music (IAEM) project which is also presented in this paper.The IAEM is intended to be a platform to access an extensive and distributed archive of electronic music. It combines collaborative tools, real time signal processing on the client side and the content of the archive to a powerful teaching, research and publishing tool.

1. INTRODUCTION

The IAEM project[1] is developed to present an extensive amount of digitised music following a new approach. It extends the capabilities of an ordinary electronic library with a collaboration platform and a audio rendering machine in order to make it an Internet based multi-media information source for students, lectures and other researchers.

There are many fields of applications possible for the system proposed. Because of the flexible design of the audio rendering machine it is possible to introduce real-time signal processing with user-defined algorithms to web based applications. Through the multi-channel streaming capability multi-track recordings can be received in their correct historical and acoustical context. This predestines the system to be used in teaching and research. However, in principal every signal processing task can be delegated to the iARS plugin.

The distributed architecture of the content databases allows the integration of electronic archives from different attending institutions. The partners share the common IAEM portal to access the data, but the databases are located at the institutions. This is a very scalable approach because the effort to migrate and to maintain the data is not centralised at the operator of the IAEM portal. It splits the efforts for hardware and bandwidth between the partners. Also legal aspects are considered by this architecture. In legal terms there is a difference whether the digital copy remains physically at the owner's location or not. Allowing distributed databases instead of one central database improves the legal certainty.

The IAEM system is also intended to be a publishing platform for the users. It allows to publish music pieces as well as algorithms for the audio rendering. It is hoped that the system will serve as a vital platform for many people contributing to its content.

[1] Home: http://iaem.at

2. THE ARCHITECTURE

The architecture of the IAEM system is a classical server-client approach with distributed databases as back-end data source. But clients may also connect to the content databases directly. Figure 1 illustrates the approach.

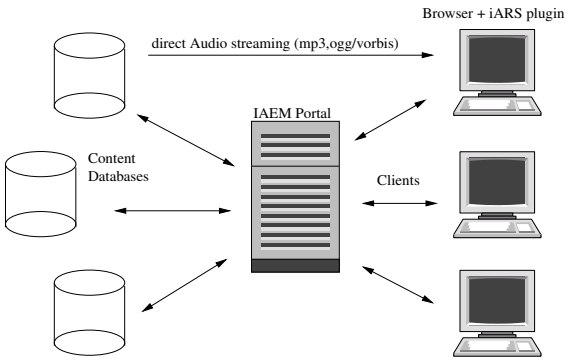

Figure 1: Basic structure of the IAEM including client terminals

The core is the IAEM portal server which provides all collaboration tools and a content management system. This portal may connect to a list of content databases where the music pieces are stored along with some additional meta-data like information about the composer, the performing orchestra etc. The portal can process search-queries on the data in order to offer it to the user via the web interface. The user can browse through the information, attend to discussions or select a music piece and a audio rendering algorithm for listening. If the user decided to receive a piece of music, the iARS browser plugin is started at the client. It loads the chosen algorithm and connects to the content database to receive the requested music piece as an audio stream. The plugin also provides a graphical user interface in the browser window. The GUI contains controls with which the behaviour of the audio rendering algorithm may be altered during operation.

The direct connection between the client and the content database decreases the hardware requirements for the IAEM portal as the audio streams do not need to be routed through the portal.

3. THE CONTENT DATABASES

A IAEM content database system consists of three main components. The database itself is storing references to the audio data in the file-system and the additional meta-data. This database can be queried by the IAEM portal through a standard SQL interface. The control block is also communicating with the IAEM portal. It is responsible for carrying out commands received by the portal via a XML-RPC interface [1]. With these commands the portal can initialise a stream, start or pause it and remove the streaming mountpoint. A standard streaming server is part of the content database system for streaming the audio data. Figure 2 illustrates the architecture.

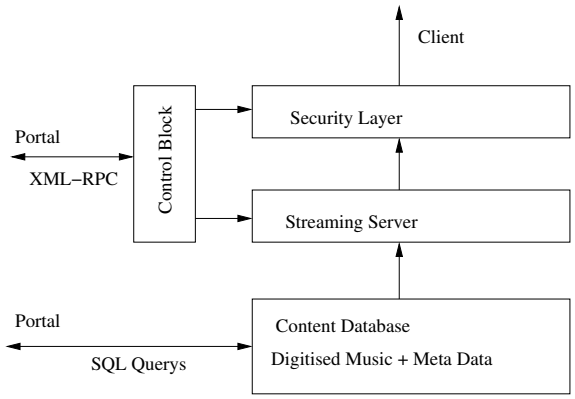

Figure 2: Content database system

In order to provide multi-channel capabilities the IAEM content database system needs to employ a streaming server technology which supports multi-channel audio formats. Ogg vorbis is a new compressing audio data format for encoding mid to high quality audio at variable bitrates from 16 to 128 kbps/channel. Since version 1.0 rc1 this standard also provides channel coupling mechanisms designed to reduce effective bitrate by both eliminating interchannel redundancy and eliminating stereo image information labelled inaudible or undesirable according to spatial psychoacoustic models [2].
Along with references to the audio data the content database contains meta-data related to the music pieces. The design is based on a relational database structure and is similar to commonly used library systems. It follows cataloguing standards to make data migration as easy as possible. The interface for portal queries is a standard SQL command set.
The control block is a simple state machine receiving commands from the portal and controlling the streaming server and a security layer. Implementations of XML-RPC are available for the most common programming languages. The standard also proposes introspection methods and multicalls.

4. THE IAEM PORTAL

The IAEM portal is a content management system with various collaboration tools and additional features to drive the iARS plugin and to query the content database systems. The chosen framework is Zope extended with CMF and Plone.
The data presented by the portal is legally sensitive so that a secure authentication method is compulsory. The Zope system provides a LDAP authentication product with which the user must log in before the portal can be used. This allows also a personalised environment with user defined folders and content. The rights can be set for every single user so that the access to music pieces can be clearly determined to prevent any legal conflicts.
The collaboration and exchange of information is supported by a set of tools including calendars, Wikis[2] and a discussion tool. The discussion tool developed for IAEM combines the capabilities of an mailing list and a discussion forum including an archive of contributions. The intention is to keep the number of tools low due to simplicity in the usage of the portal, while providing the most important features of commonly used collaboration tools.
Information digging is supported by a normal single line search and a more powerful advanced search facility. Another way to retrieve information is to use personal *Information Agents*. Agents reflect a certain interest of the user and try to collect relevant information. Users may create several agents with very different parameter settings, each representing a topic of interest for the user. The agents search for collaborations tools, music pieces and other meta-data and present the collected information to the user.
For publishing the portal also provides uploading to a content database. The access rights for user published data can be set by the author via the portal.

5. THE iARS BROWSER EXTENSION

iARS (internet Audio Rendering System) is a browser plugin extending the browser's capabilities with a flexible audio rendering machine. It can be invoked by an HTML "object" tag within web pages. The signal processing is done by the Pure Data[3] which is launched by the plugin and remote controlled via a XML-RPC interface. The algorithm processed can be defined as a regular Pd patch along with a graphical representation of the patch. This is done using a IDL (interface description language) introduced in section 5.4. According to this description the plugin draws controls into the browser window with which the behaviour of the algorithm can be altered during operation. Figure 3 shows the iARS plugin embedded in a browser window.

5.1. Operation

The plugin is launched by using the "object" tag embedded in regular HTML code. A MIME type (*application/iARS*) is registered by the plugin at the browser which refers to the data type associated. Whenever a object tag is referring to this MIME type the browser starts the iARS plugin (available as a shared library object). A window handle provided by the browser is assigned to the plugin for its graphical representation. The object tag includes three parameters determining the data sources. A typical tag looks like:

Listing 1: Embedded object tag

```
<html>
 ...
<body>
 <object type="application/iARS"
   width="400"
```

[2]Wiki is an open editing system that allows multiple users to work on the same web content. It supports a history and reviewing processes. `http://www.wiki.org`

[3]Pure Data by Miller Puckett `http://crca.uscd.edu/~msp`

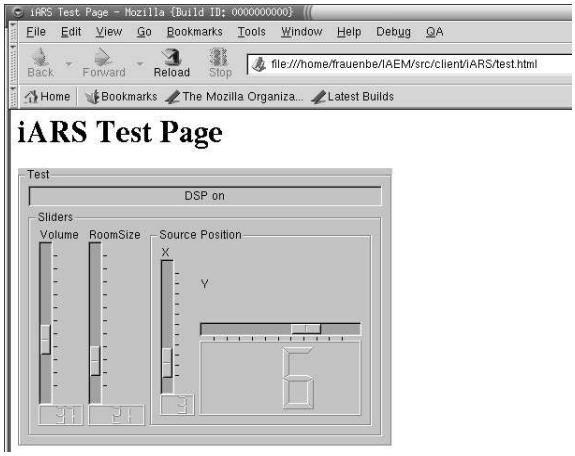

Figure 3: Screenshot of the iARS plugin embedded in a browser window

```
  height="300"
  <param name="PATCH"
      value="http://iaem.at/test.pd">
  <param name="GUI"
      value="http://iaem.at/gui.xml">
  <param name="STREAM"
      value="http://content.at/AS52SA.ogg">
 </object>
...
</body></html>
```

The patch parameter refers to the implemented algorithm to be used as a URI, the GUI parameter to the corresponding IDL file determining the graphical representation of the patch. The files will be requested by the plugin and must be available from the server. The stream parameter provides information about the audio data source. It tells the plugin to which server to connect and the name of the streaming mountpoint. For security reasons the mountpoint name is randomly generated by the portal for each session. This secret name is distributed only to the content database system and the requesting browser so that no other user may have unauthorised access to the streamed data.
For future versions of iARS a more complex security concept is planned employing certificates for authorised clients and content databases. With this the peer authenticity will be determinable unambiguously and the connection may be encrypted.

5.2. Components of iARS

The collaboration of iARS and the hosting browser is ruled by the Netscape plugin API[3]. This API determines the way plugins are called and their entire life-cycle. Considering that life-cycle and the requirements for operation, the following components can be identified (as shown also in figure 4).

- The plugin API module
- Initialisation task module
- Central controlling module
- Graphical output
- Pd communication (XML-RPC) module

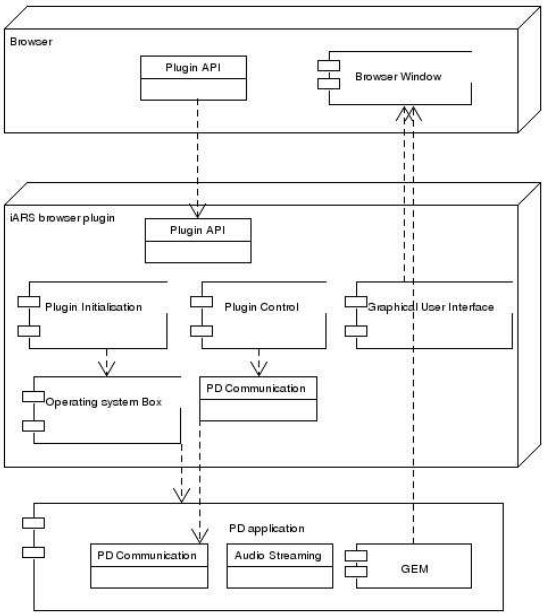

Figure 4: Deployment diagram for the iARS browser plugin

5.3. Pure data

Pure Data is a real time signal processing tool for PCs available for different operating systems like Linux and Windows[4]. There are many extension libraries available for Pd since Pd provides a simple interface to extending its capabilities. The two main extensions developed for the IAEM project are the XML-RPC interface and the improved ogg vorbis streaming external. The main advantage of using Pd as the processing core application is that there already exist many patches. The generic approach of the plugin allows to reuse these patches only with minor adjustments.
Ogg/vorbis is a standard for streaming compressed audio over the net. Support for ogg/vorbis in Pd was basic when IAEM was launched. Streams could only be stereo and there was a significant threading problem causing klicks. The `pdogg` library initially written by Olaf Matthes[4] was extended and threading was improved so that the external now is capable of transmitting and receiving multichannel streams in high-quality. The implementation is based on the reference libraries available from the ogg/vorbis website[5].
The XML-RPC interface to the Pd program is intended to become a comfortable standard of remote controlling the application. It is possible to load and close patches, but also to communicate with elements of a patch. It replaces the `netsend` and `netreceive` commands with a more flexible and powerful communication interface.

5.4. Graphical representation

The graphical representation of a patch is not defined from the data within the patch as Pd's GUI is presenting it. We have chosen to use an extra file to determine the graphical representation in the

[4] `http://www.akustische-kunst.org/`
[5] Available from `http://www.vorbis.com/download.psp`

browser window because the original way seemed to be too complex and technically orientated. The target user group may not be familiar with Pd and may be easily confused by its complexity. This demanded the definition of an interface description language (IDL) more suitable for our application than the existing. The implementation of the controls was made using Trolltech's Qt toolkit [5].

Within the IDL file several controls are defined which are bound to elements of the Pd patch. If either the user interacts by changing the value in the GUI or the patch alters the value the counterpart is informed. So, parameters of the patch can be altered and values can be displayed correctly. The IDL file is in XML format and fully defined by the document type definition "idl.dtd" containing all possible tags and their relations. The following listing shows an example of a IDL file describing a graphical representation of a Pd patch.

Listing 2: XML IDL example

```
<?xml version="1.0"?>
<!DOCTYPE interface SYSTEM "idl.dtd">
<interface>
 <author>Christopher Frauenberger</author>
 <patch>Ambisonic 3D</patch>
 <version>1.0</version>

 <input name="oggstream"/>

 <group name="Test" orientation="vertical">
  <onoff name="DSP on" bind="dspon" value="1"/>
  <group name="Sliders"
    orientation="horizontal">
   <vslider name="Volume" bind="volume"
     min="0" max="100" value="37"/>
   <vslider name="RoomSize" bind="size"
     min="0" max="100" value="21"/>
   <group name="Source"
     orientation="horizontal">
    <vslider name="X" bind="x"
     min="0" max="20" value="3"/>
    <hslider name="Y" bind="y"
     min="0" max="20" value="6"/>
   </group>
  </group>
 </group>
</interface>
```

The listing above results in the interface controls shown in figure 3. Necessarily, a IDL file must determine an input receiver. This receiver will be fed with the information of the *stream* parameter provided in the object tag in order to connect to the appropriate streaming server. The IDL DTD also allows grouping of controls. The *group* tag provides a simple layout parameter determining whether the controls will be placed horizontally or vertically. Each control must define a binding parameter determining the receiver object in the Pd patch and a legal value range along with the current value. This makes these IDL files also suitable for saving the current settings into a user file. For example, if a user have found his favourite mixer settings for a special music piece he might save this settings in his private folder for later use.

6. CONCLUSION

The proposed system combines very recent technologies in the field of signal processing and Internet tools to a powerful research and lecturing tool. All components were designed to be as flexible and generic as possible. The distributed architecture allows different partners to collaborate for providing their clients a comprehensive library of electronic music.
The iARS plugin is an approach to introduce real-time audio rendering to the world of web applications. The underlying Pd program was chosen because of its performance and availability for a wide range of platforms.

7. ACKNOWLEDGEMENTS

This project was kindly funded by the Austrian Federal Ministry for Education, Science and Culture within the "New Media in teaching at universities and polytechnics in Austria" project framework.

8. REFERENCES

[1] D. Winer, "Xml-rpc specification," Tech. Rep., Userland, xml-rpc.com, 1999, http://www.xmlrpc.com.

[2] Ogg/Vorbis, *Ogg Vorbis I format specification: introduction and description*, 2003, http://www.xiph.org/ogg/vorbis/doc/vorbis-spec-intro.html.

[3] Netscape, *Netscape Geck Plug-in API*, 2002, http://devedge.netscape.com/.

[4] Miller Puckette, *Pd Documentation*, 2003, http://crca.ucsd.edu/~msp/.

[5] Trolltech Inc., *Qt Reference Manual*, 2003, http://doc.trolltech.com/3.1/.