Internet Archive of Electronic Music IAEM internet Audio Rendering System iARS

Christopher Frauenberger[1] and Winfried Ritsch[1]

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

Abstract. The Internet Archive for Electronic Music (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 with the concept of learning sequences to a powerful teaching, research and publishing tool. The internet Audio Rendering System (iARS) refers to a client browser extension which is part of the IAEM system. It extends a web-browser with a flexible real time audio processing capability supporting multi-channel processing. This enables users of the system to perceive multi-track recordings in their correct acoustical context. Through its built in user interface iARS is capable of presenting mixing devices and graphical scenes creating interactive virtual environments. These environments may be used embedded in learning sequences in teaching or for experimenting with new music algorithms.

1 Introduction

The IAEM project is intended 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 a 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. Additional graphical scene rendering allows to create virtual concert situations with an interactive mixing device.

The distributed architecture of the content databases allows the integration of electronic archives from different attending institution. 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 also splits the efforts for hardware and bandwidth between the partners.

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 the content.

Offering music for listening in the Internet must consider legal issues to guarantee the legal certainty. The operator of each content database is responsible for the content he provides. He must be authorised for digital copying the source and publishing it for a certain user group. The restrictive authentication mechanism in the IAEM system allows to set up different access rights for various user groups.

2 Architecture

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

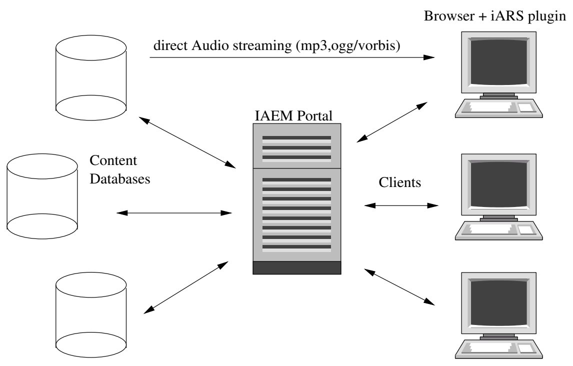

Fig. 1. Basic structure of the IAEM including client terminals

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 metadata as usual in common music libraries (composer, artists 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 his or her browser. 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 the behaviour of the audio rendering algorithm online (means during operation).

The direct connection between the client and the content database decreases the hardware requirements for the IAEM portal. If every stream connection would be routed via the portal, the available andwidth would restrict the number of connections.

3 System Requirements

Due to the distributed architecture illustrated above the requirements for the single components of the system are not very high. The content databases need to have sufficient storage space for the archive intended to hold. The Internet connection needed depends on how many users are expected (and authorised) to request music pieces from the database. However, a 100Mbit/sec (T1) connection usually available at institutions willing to share their archive is enough to serve a reasonable number of clients (theoretically up to 800 for 128kbps streams).

The portal runs a content management system including a database backend. For this component too the requirements are not very high. A customary server provides usually sufficient performance. The prototype built at the IEM Graz is running a 2MHz Pentium XEON with a 4.6GB RAID system.

The iARS plugin is based on the Pure Data programme and requires a running installation on the clients PC. Fortunately Pure Data supports a wide range of platforms like Windows or Linux. iARS is written for Browsers supporting the Netscape Gecko Plugin API [1] (Netscape 4.7, 6x, 7x, Mozilla 1x).

4 The Content Databases

An IAEM content database system consists of four 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 [2]. With these commands the portal can initialise a stream, start or pause it and remove the streaming mountpoint. It also controls the security layer which is by now only a future concept. It should strengthen the security and peer authenticity by certificates and SSL tunnel transmission.

The migration of audio data into the system is under the responsibility of the operator and/or the partner institution. The IAEM system provides a good reason to digitise old music pieces and to migrate even multi-track recordings. The multi-channel and audio rendering capabilities of the system make a realistic reproduction of such pieces possible.

4.1 Streaming

n 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.

For supporting both the newer ogg vorbis format and mp3 the chosen streaming server is IceCast 2. It is fully controlled by the control block and sets up mountpoints with specific names. These mountpoint names are random strings generated by the portal and sent to both, the streaming server and the iARS client. This provides additional security because only the one client who requested the streaming is aware of the name of the mountpoint.

4.2 Database

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, but simplified to suit our requirements. The interface for portal queries is a standard SQL command set. Meta-data include references to a composer database, lyrics, scores and other analysing remarks.

5 The IAEM Portal

The IAEM portal is a content management system with various collaboration tools with features to drive the iARS plugin and to query the content database systems. The chosen framework is Zope [3] extended with CMS[1] and Plone [4]. For integrating learning sequences into the portal eduPlone was chosen as the state-of-the-art in the field.

The data presented by the portal is legally sensitive so that a secure authentication method is compulsory. The Zope system provides a LDAP[2] 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.

Collaboration tools are integrated to ease the communications between the users. It is hoped that vital discussions and information exchange will contribute to the content of the portal. Mailing-lists, discussion forums, information agents

[1] Content Management System
[2] Lightweight Directory Access Protocol

and other common collaboration tools are available to make such exhanges possilble.

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. For searching the content databases a single line search is implemented as well as a more complex advanced searching facility.

The didactic value for teaching is given by the integration of the tools into eLearning courses. eduPlone is a product especially designed for the use with the Plone CMS system and provides learning sequences for designing eLearning courses [5]. For example: lecturers may design a course for digital recording techniques providing raw material (a piece of not post-worked music) via the portal. In various learning sequences the students are intended to solve problems like finding a proper loudspeaker arrangement for the recording. They could play around with the iARS system and save their settings to submitt their results. Along such sequences a vital exchange beyond the students and the lecturer can take place employing the collaboration tools integrated into the system.

6 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 "object" tag within web pages. The signal processing is done by the Pure Data programme 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). According to this description the plugin draws controls into the browser window with which the behaviour of the algorithm can be altered.

iARS implements the Netscape Gecko Plugin API to communicate with the browser. During the initialisation process the plugin checks for running instances of Pd and launches an instance if needed. The plugin control block is remote controlling the Pd programme and builds the graphical representation of the patch loaded. The Pd programme is launched with externals which extend the capabilities of Pd for XML-RPC communication and audio streaming. The GEM library is used to draw real time computer graphics to an assigned window area using openGL.

6.1 Operation

The plugin is launched by using the "object" tag embedded in regular HTML code. A MIME type is registered by the plugin at the browser which refers to the the data type associated. The following listing shows an example HTML code for embedding iARS objects.

Listing 1.1. Embedded object tag

```
<html>
 ...
<body>
<OBJECT type="application/pd"/>
 <param name="patch" value="http://iaem.at/amb.pd">
 <param name="gui" value="http://iaem.at/ambgui.xml">
 <param name="stream" value="http://db1.at:8888/NHS271/">
 <param name="extras" value="http://iaem.at/extras.zip">
</OBJECT>
 ...
</body></html>
```

The object's application/pd MIME type causes the browser to launch iARS. A window handle provided by the browser is assigned to the plugin for its graphical representation. The "data" field determines the URI of the requested audio data, "patchsource" and "gui" both in a URI format assign the Pd patch to be loaded and its graphical representation. An zip archive of extras may be specified to provide the patch with additional abstractions or Pd externals. The Gem external of Pd allows the plugin to draw virtual concert situations into the browser window. It can be used, for example, to show the position of virtual surround loudspeakers and even allow to alter their position in the virtual room.

6.2 Pure Data

Pure Data is a real time signal processing tool for customary PCs [6]. There are many extension libraries available for Pd extending its capabilities. The two main extensions developed for the IAEM project are the XML-RPC interface and the 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.

The XML-RPC interface to the Pd programme 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 every single element of a patch. There are mechanisms to bind callback functions to symbols so that a event triggered communication desired for GUIs is possible.

6.3 Graphical Representation

The graphical representation of a patch is not defined within the patch. This allows the reuse of existing patches and the definition of a interface description language (IDL) more suitable for our application than the existing. The implementation of the controls was made using Trolltech's Qt toolkit [7].

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 following listing shows an example of a IDL file describing a graphical representation of a Pd patch.

Listing 1.2. Interface Description Language 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="stream"/>
<group name="Example" orientation="horizontal">
  <vslider name="Volume" bind="volume"
           min="0" max="100" value="10"/>
  <levelmeter name="Left" bind="vul"
              min="-100" max="0" value="0"/>
  <levelmeter name="Right" bind="vur"
              min="-100" max="0" value="0"/>
  <group name="GEM" orientation="vertical">
    <onoff name="GEM Window" bind="gemwindow" value="1"/>
    <onoff name="OpenGL" bind="draw" value="1"/>
    <hslider name="Rotation"
             bind="rotate" min="0" max="180"/>
  </group>
</group>
</interface>
```

In this example a simple interface is built with a vertical slider for the volumem, two levelmeters, another slider along with two buttons. The group tag allows the user interface elements to be grouped together in a frame. The description shown above results in a graphical user interface shown in figure 2.

All possible tags and their relations are described in the document type definition “idl.dtd”.

7 Conclusion

The proposed system combines very recent technologies to a powerful research and lecturing tool. All components were designed to be flexible and generic. 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 programme was chosen because of its performance and availability for a wide range of platforms.

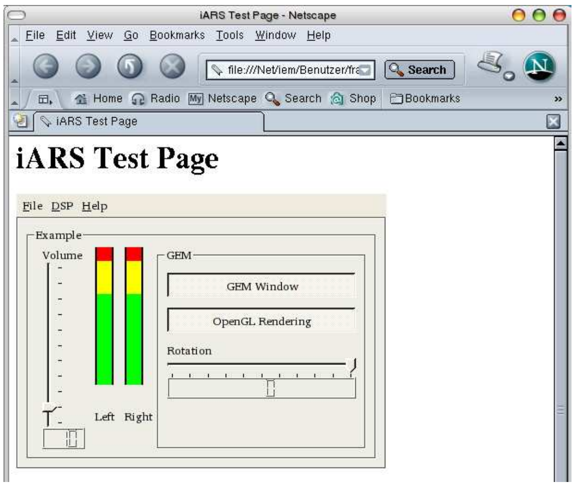
iARS Test Page - Netscape
File Edit View Go Bookmarks Tools Window Help
file:///Net/iem/Benutzer/fra
Search
Home Radio Netscape Search Shop Bookmarks
iARS Test Page
iARS Test Page
File DSP Help
Example
Volume
Left Right
GEM
GEM Window
OpenGL Rendering
Rotation

Fig. 2. Screenshot of the interface described by the IDL example above

Future work will definitely need to proof the concept by usability tests. The portal and its components will be redesigned on the basis of the results of such studies.

8 Acknowledgement

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.

References

1. Netscape, *Netscape Geck Plug-in API*, 2002, `http://devedge.netscape.com/`.
2. D. Winer, "Xml-rpc specification," Tech. Rep., Userland, xml-rpc.com, 1999, `http://www.xmlrpc.com`.
3. Zope, *The Zope Book*, 2004, `http://zope.org/Documentation/Books/ZopeBook/2_6Edition/`.
4. Plone, *The Plone Book*, 2004, `http://plone.org/documentation/book`.
5. "eduplone concepts," `http://www.eduplone.net/concepts/`, 2004.
6. Miller Puckette, *Pd Documentation*, 2003, `http://crca.ucsd.edu/~msp/`.
7. Trolltech Inc., *Qt Reference Manual*, 2003, `http://doc.trolltech.com/3.1/`.