Sociedade de Engenharia de Áudio

Artigo de Congresso

Apresentado no 8º Congresso de Engenharia de Áudio
14ª Convenção Nacional da AES Brasil
4 a 6 de Maio de 2010, São Paulo, SP

An auralization engine for open mobile phones

Regis Rossi Alves Faria[1] and Gianpaolo Evangelista[2]
[1]Laboratório de Sistemas Integráveis – Universidade de São Paulo
São Paulo, SP, 05508-010, Brazil
[2]ITN – Linköpings Universitet
Norrköping, SE-60174, Sweden
regis@lsi.usp.br, giaev@itn.liu.se

RESUMO

Este artigo apresenta os primeiros resultados de portar uma máquina de auralização existente para um celular de *hardware* e *software* abertos. Até então a manipulação de cenas sonoras para propósitos de auralização tem sido realizada só sobre plataformas fixas. Entretanto, o aumento da capacidade computacional e recursos de áudio em portáteis rapidamente estimula a migração de tais funções para este domínio. Ilustramos o desenvolvimento de uma aplicação de auralização distribuída controlável a partir da plataforma portátil, onde o usuário pode controlar a produção e a reprodução de uma cena sonora. O celular escolhido possui uma tela sensível ao toque e oferece a vantagem de programação para um núcleo Linux. Os resultados obtidos mostram novas perspectivas para aplicações de áudio em dispositivos portáteis.

ABSTRACT

This paper presents the first results of porting an existing auralization engine into a mobile phone with open-source hardware and software. So far the manipulation of sound scenes for auralization purposes has been performed only in desktop platforms. However, the increase of processing power and audio resources in mobile phones rapidly encourage the migration of such functions into this domain. We illustrate the development of a distributed auralization application controlled from within a mobile platform where a user can control the spatial audio rendering and reproduction of a sound scene. The chosen phone has a touch-screen interface and offers the advantage of programming to a Linux OS kernel. The results show exciting new perspectives for mobile audio applications.

0 INTRODUCTION

As mobile platforms get more powerful and widespread, functions once confined to desktop systems progressively migrate into these devices with additional advantages: they are portable, have several communication ports, and are appealing to personalizing services and applications.

Spatial auralization in mobiles is a new field of applications. Despite the fact that current mobile devices do not have more than two speakers, they are already powerful processing units, useful in many sound applications. These include, for example controlling auralization processes running in other computing nodes and monitoring spatial sound fields locally, either through

the use of an external high-fidelity soundcard connected to its output or through its own loudspeakers (even though there are yet severe limitations in terms of irradiation, volume and frequency response).

This scenario leads to a new concept of using mobile phones as processing units in a chain of distributed auralization, together with other usual platforms, such as PC-based audio workstations. We believe that mobile auralization tools are yet undiscovered and handy to tackle many professional problems in the audio chain, such as distributed audio ingest, mixing and monitoring problems.

This paper approaches the programming of an auralization engine into an open-source phone, and is organized as follows: Section 2 describes some relevant and innovative aspects of this proposal, and presents briefly the adopted auralization engine architecture, its software, and the mobile phone used. Section 3 addresses the system design and implementation, covering a functional description, the system components, and reports the porting the auralization engine into the mobile phone framework. Attained results are presented and commented in Section 4. Section 5 concludes the paper with final considerations and an overview of next steps.

1 SYSTEM PROPOSAL

This work proposes porting the existing AUDIENCE auralization engine [1,2] into an open-source hardware and software mobile platform. The mobile platform chosen is the Openmoko™ [4], with both the hardware and operating system released as open source, and with several unique features[1]. The auralization engine is an adapted version of the OpenAUDIENCE software [3]. With this system one can customize spatial sound scenes with multiple sound sources, and control their auralization directly from the mobile phone in a wide range of applications.

The proposed system has many innovative features, linking areas that do not have a previously defined confluence with advanced audio processing. Until now no mobile platform addressed the customization of auditory scenes in real time. The insertion of this new paradigm for mobile audio computing can foster new ways for musical design and sound manipulation in different types of environments. There are many new areas of applications for this concept in the domains of professional and consumer audio and musical performance, among others.

To validate this concept, a *distributed auralization application* was built and tested. The initial scope is basically to control an auralization session from the mobile phone, and to have the audio rendering take place in a remote desktop computer feeding a loudspeaker rig. This scheme is illustrated in figure 1 and the expected sequence of events is elaborated below.

An auralization applicative is accessible as an icon in the mobile applications desktop. The user double-clicks it to start the program, and then open a specific auralization session and its corresponding sound scene. He/she can then manipulate the sound scene directly from the mobile unit. The actual sound rendering (from the objects' sounds) and the spatial audio coding and decoding to a specific multichannel loudspeaker setup are all accomplished in the remote desktop computer.

[1] Openmoko is a trademark by Openmoko, Inc.

The application built is based on the Pure Data graphical programming environment (Pd) [5,6], and consists of a master patch connecting processing blocks. The concept is well known and used in many signal processing software packages. Instantiating blocks and connecting them is a matter of a few graphical-oriented actions the user can take in real time. By doing so, one customizes the application, inserting desired functions or eliminating others. For example, transport sliders and buttons to calibrate loudness and to mute sounds can be instantiated and put to work on-the-fly.

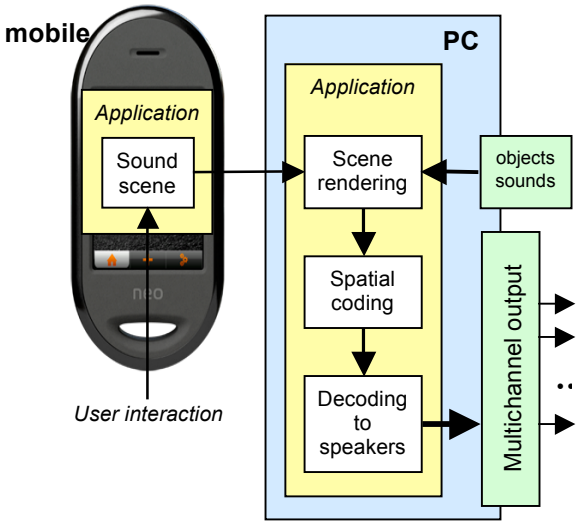

Figure 1 Scheme of a mobile-controlled distributed auralization session with 2 nodes: the mobile phone and a remote PC

1.1 Usage

In many practical situations it is desired to establish and to control an auralization session operating from a portable phone. Examples include applications where multiple users can interact in immersive shared audiovisual environments, controlling distinct sound sources performing in the same scene, such as in games and professional auditory installations.

Our test application consisted in an auralization session of a sound scene using the mobile phone as the controller node and a remote desktop computer (a PC) as the sound renderer. The sound scene consists of a square area interface on the screen mapping an open-field virtual volume enclosing from 3 up to 6 sound sources plus the listener. The user can freely move any sound source and the listener within the virtual area while coordinating the auralization events flow, such as starting events, muting or turning on sounds, modifying their reference loudness, and changing spatial encoding and decoding parameters.

The session itself can be started and controlled from the mobile interface. After starting, it establishes a communication link between the two nodes and a chain of processes both locally (on the mobile unit) and on the remote computer. It is then possible to manipulate the given sound scene with all its sound objects, to control sound source parameters and how the virtual sound scene is rendered in the real audition area, including altering the multichannel reproduction mode.

All these functions are accessible in different layers of the proposed system. Moreover, if more mobile units are

available, the session control may be shared and the units may control different sound objects in the same scene.

1.2 The AUDIENCE auralization system

The AUDIENCE is an auralization system based on a modular processing architecture oriented to the main functions in the spatial sound field encoding and decoding chain. It has been successfully used in immersive CAVE virtual reality and sound-scene-oriented spatial music applications [2,7,8].

The architecture addresses four functional layers which cluster the major functions required for the production and the reproduction of spatial audio, as shown in Figure 2. This subdivision has been proved advantageous to systematically separate the tasks of sound scene direction, spatial audio rendering, and multichannel display.

Each function must be properly implemented in one of the layers of this system. Briefly, the layer 1 addresses the production of a sound scene, the layer 2 renders its acoustical model, and then it is temporally and spatially encoded in the layer 3, using any suitable spatial audio encoding method. It is then decoded by layer 4 into a desired set of signals (e.g. stereo, quadraphonic, planar speaker rings, 5.1, 6.1, etc.). Between layer 3 and layer 4 there is a delivery layer, not addressed in the architecture.

This approach has several advantages from the viewpoint of implementation, by decoupling the main processes in the chain, and allowing them to be performed by independent modules (in software and/or hardware). Also, a spatial auditory scene is represented by its objects and their relations, and not by a fixed multichannel sound set attached to a specific listening mode. For this, the AUDIENCE can be considered sound-scene-oriented.

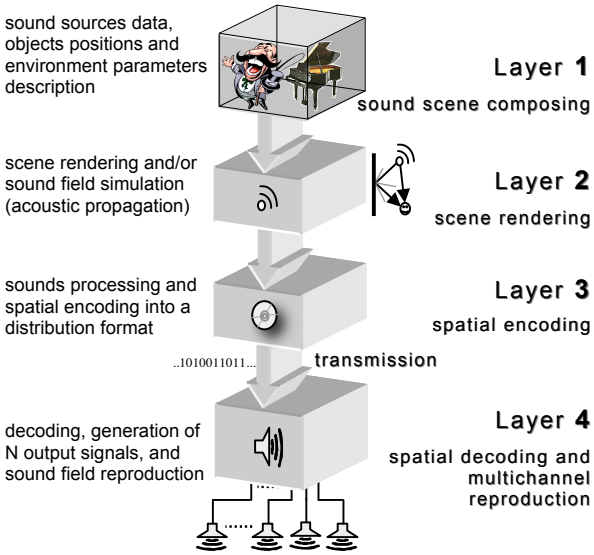

Figure 2 The AUDIENCE 4-layer architecture for auralization

The OpenAUDIENCE is a project aiming at the collaborative development of software tools for immersive and interactive auralization [3]. Its software platform consists of an integrated auralization environment, which includes interactive audio tools for scene description and manipulation, for acoustic rendering, and spatial and multichannel audio codec's (e.g. ISO/IEC MPEG-4 AAC, Ambisonics and MPEG Surround). Its current version is based on libraries and patches for the Pure Data (Pd), implemented in Win32, Linux and MAC OS operating systems.

1.3 The open-source platform Openmoko

The Openmoko project is one of the first initiatives to implement a hardware platform for communication that uses the concept of open source, and thus shares common ground with the OpenAUDIENCE project and with the Pd programming language. Its mobile platform is an open-source GSM phone issued under GPL and LGPL that uses a Linux kernel, hence offering a friendly programming environment.

Since 2007 a company joined the project and began producing the hardware, which has its technical specifications disclosed in the project website [9]. Some of its most relevant features are shown in Table 1.

Table 1 Openmoko mobile phone features (GTA02) [5]

ARM 400MHz processor w/ 2D/3D graphics acceleration	128MB SDRAM and 256MB NAND Flash memories, and microSD expansion slot
Touch-screen (VGA 480x640)	Wi-Fi (801.1 b/g)
GPRS (2.5G) and Bluetooth 2.0	GPS and 3-axis motion sensors
Microphone and stereo output, counting on a codec chipset of low THD (-84dB 'A' weighted@48kHz)	GSM tri-band (850 or 900 MHz)

The processing speed is sufficient to run audio applications concurrently with communications sessions and a high-resolution graphical display.

In the current Openmoko programming environment, application-oriented libraries run over a set of high-level and low-level services available in the framework. These all run on top of a Linux kernel [4].

The software development framework includes only free libraries, including those created by the Openmoko main team, which are:

- *libmokocore*: addressing the message exchange between applications, and reading/writing data of persistent applications
- *libmokonet*: which manages which communication devices should be sent information
- *libmokopim*: which is a high-level API library for PIM data (personal information management)
- *libmokoui*: responsible for the management of audio and video of the phone.

2 DESIGN AND IMPLEMENTATION

This first implementation is part of a proposed roadmap of developments towards an integrated environment for mobile audio computing. It aimed at porting the

OpenAUDIENCE into the Openmoko so that this could be used as one node in a distributed auralization task. It should then run processes (functions) of any of the four layers of this architecture, and perform some or all functions required in one auralization chain.

2.1 Functional design

The design of the application modules in the AUDIENCE framework must meet the requirements of the architecture, clearly identifying in which functional layer each component of the system falls. Also, the specification of input and output signal formats and data structures must be compatible with the interfaces allowed for each layer, as presented in [1].

Table 2 depicts which functions were designed for implementation in each layer.

Table 2 Openmoko mobile phone features (GTA02) [5]

Layer	Features
Layer 1	The scene graphical user interface (GUI) – holds the description of the sound scene and parameters of the sound objects. It is manipulated directly by the user from the touch-screen.
Layer 2	Acoustic model rendering and effects processing – includes image-source simulation for square rooms [2].
Layer 3	Spatial encoding block – encoding of the audio signal in spatial and/or multichannel formats (e.g. Ambisonics, AAC, etc.). Used codecs' profiles and parameters are input as arguments in their blocks (e.g. Ambisonics order: changing the Ambisonics order in real time is possible and this implies establishing a different number of internal channels).
Layer 4	Spatial decoding and playing blocks – decoding of spatial/multichannel audio and playing to local loudspeakers. Sound object transport windows, with volume slider and mute button. Decoding blocks match the profile and parameters used in the correspondent layer-3 blocks.
Communication auxiliary layer	Sending/reception of messages to/from remote patches via a TCP/IP transport – uses Pd *netsend* and *netreceive* commands over wireless (e.g. Bluetooth, Wi-Fi) and USB protocols.
I/O auxiliary layer	Control of audio input and output – access objects sound files stored in memory and playback [2].

Figure 3 illustrates how the target spatial audio engine fits within the protocol stack of the Openmoko operating environment.

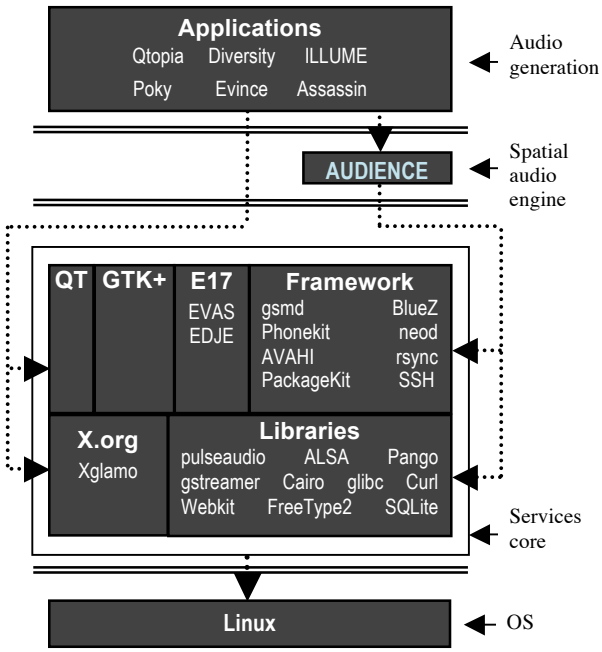

Figure 3 AUDIENCE integrated into the phone software stack

Instead of directly accessing the audio resources of the operating system services core, applications generating audio can use the AUDIENCE engine as an intermediate layer to render a customized auditory environment shared by multiple sound sources. By doing so, the AUDIENCE system is integrated into the Openmoko software architecture as a tool for customization of auditory environments.

2.2 System implementation

The implementation of the mobile auralization engine was accomplished by customizing the OpenAUDIENCE software for use in the Openmoko platform, which involved adapting the layer-1 GUI component and building auxiliary patches for the communication between the portable device and the PC. Then we looked after the effective functioning of the integrated Openmoko-PC solution.

A distributed auralization session is established with at least two nodes: one mobile unit and the remote PC. Figure 4 shows the layout of the distributed auralization system. It gives an overview of all functional blocks both in the mobile and in the PC nodes, including the hardware components.

[2] Not used in this implementation.

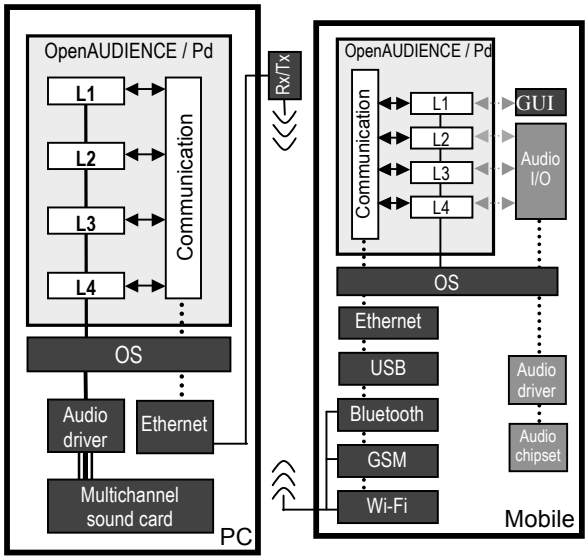

Figure 4 Block diagram for the distributed auralization system

On both nodes, OpenAUDIENCE library runs on top of Pd environment, which by its turn accesses the OS services, including the audio I/O resources and Ethernet-based communications. In the mobile there are several choices for transport layers, as shown in the Figure 4.

Some challenges were faced to overcome limitations of processing power and memory. Openmoko operates at 400 MHz and has an ARM CPU architecture, which implies a different compiling strategy. Initially we emulated Openmoko for several tests, and then started working on the real platform. An open-source processor emulator (QEMU) was used as auxiliary tool to facilitate user's interfacing and debugging in a PC environment, with less storage limitations [10].

Porting the OpenAUDIENCE Linux version to Openmoko required first installing the Pd environment in the phone, re-compiling all components, and then fitting a functional package in memory.

For this, it was necessary the installation of specific development packages, because the portable device is not distributed with all required tools for mathematical processing and Linux application compiling. This made possible, for instance, customizing the graphical display blocks and the user interaction component of the OpenAUDIENCE (audce_L1_gui), which is written in C.

The final executable applicative based on the adapted OpenAUDIENCE was named AMob – a short for AUDIENCE Mobile – and was added to the collection of applications of the phone.

3 RESULTS

Figure 5 shows pictures of the implemented AMob auralization application running on top of Openmoko. The left picture shows how the application icon is accessible in the graphical desktop.

In the middle picture, the sound scene view is shown, with 3 sound sources and the listener (receiver object), indicated by the small head icon. This graphical view implements a layer 1 sound description function. The sound objects can be positioned anywhere in the defined scene representation with respect to the listener icon position by dragging their icons around with the pointing pen or using one finger on the touch-screen GUI. The listener icon can also be repositioned anywhere in the scene, thus affecting how the user will perceive the sound from each source in the scene.

In the right picture in Figure 5, another view is shown where the user can access the AUDIENCE components in layers 2, 3 and 4, through which it is possible to calibrate the relative loudness of each sound source, and to specify spatial audio encoding and decoding parameters of the auralization patch.

The auxiliary communication with the remote computer in charge of the audio rendering and multichannel playback is transparent to the user.

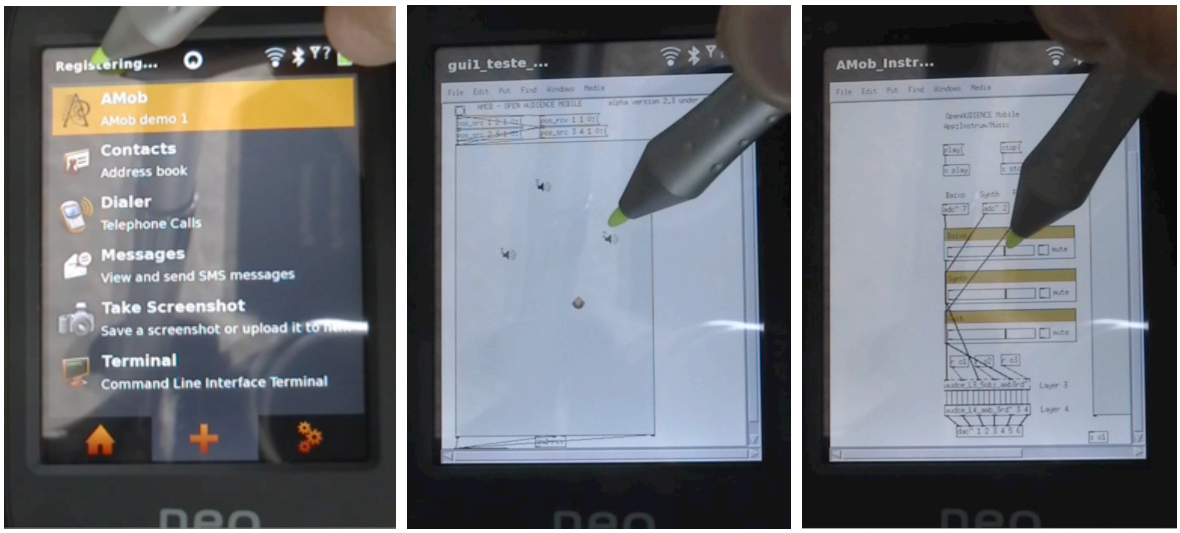

Figure 5 Pictures of the application running on the Openmoko. Left: desktop applications view showing the "AMob" icon. Center: sound scene view (layer 1 GUI). Right: auxiliary controls view (layers 2 through 4 functions)

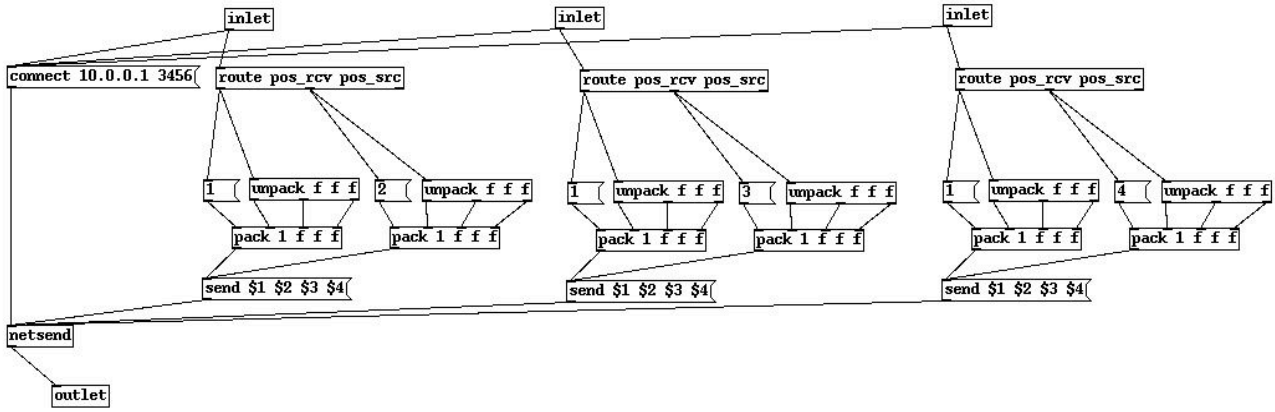

Figure 6 Auxiliary patch used for packing and sending messages between the nodes of the auralization session

In the implemented distributed auralization, the layer-1 scene control (the scene GUI) and the layer-4 reproduction control (transport commands and volume control) can be performed in the mobile node. Users interact with the sound scene using the layer-1 GUI block accessible through the touch-screen. This block sends messages to the remote patch running in the PC node with a multichannel sound card, where layers 2 through 4 operate. In the OpenAUDIENCE framework the user must select a spatial audio codec to build the auralization processing patch. In most tests, we used layer-3 and layer-4 Ambisonics 3rd order coding and decoding, with a hexagonal loudspeaker rig as output mode (6.0).

The auralization session in the Openmoko starts in few seconds. To control the process, it establishes a TCP/IP connection to the remote patch. The patch at the mobile phone sends low-bit rate command messages to its counterpart in the remote PC. Figure 6 shows a simple filtering scheme used in Pd for treating sound source positioning messages received from layer-1 on the mobile, and packing them to send to the remote patch. In the remote side, messages are unpacked and routed accordingly to the destination layers blocks. They ultimately convey the sound objects parameters to the processing blocks of layers 2 through 4.

Communication connections can be established using any available port in the phone. In our tests the Bluetooth port revealed itself effective and suitable for rapid development, although its usage is limited to a near field area.

In experimental tests, a slight delay was observed as objects were moved, indicating the need for further optimization, possibly using other graphical libraries than the Tcl/Tk [11]. The performed tests indicate that processing power and memory are still major limitations in mobiles to be used as full auralization nodes.

Although the OpenAUDIENCE software permits a wide range of possible spatial codecs and output modes, from monaural up to dodecahedron geometries, we have used Ambisonics coding/decoding up to the 3rd order, and a hexagonal ring with 6 loudspeakers for listening.

The maximum number of sources simultaneously active in a session as well as the attained immersion degree and sound quality will be limited only by the performance capabilities of the desktop node, as long as the mobile units operate only with layer-1 functions and not rendering sound locally. We have successfully tested sessions with up to 10 simultaneous sound objects with no problems and a high spatial realism delivered.

Although we have not assessed visual limits for practical operation, a higher number of objects populating the GUI might turn it more difficult to select one out of many and move it over the screen. Additionally, the maximum resolution of the touchscreen sensor combined with the model scale used (pixel to meters) may impose a minimum resolution per motion step in the virtual environment.

4 CONCLUSIONS

An auralization engine was ported into the software framework of an open-source mobile phone. In this development, we customized the OpenAUDIENCE system for use in the Openmoko portable device. A test application was then implemented, capable of controlling a distributed auralization session with 2 nodes in real time (see Figure 4). The approach is scalable, allowing two or more users coordinating sound objects in the same sound scene to operate concurrently from two or more phones running the application, and linked to the same remote PC.

This present prototype for an AUDIENCE-mobile version can already ease the creation of applications with spatial audio and the operation of sound scenes in the Openmoko. It provides a computing library for rendering the spatial sound of audiovisual scenes for applications where a sound scene with multiple objects producing sounds exist.

Although a complete stand-alone auralization through a 2-channel output is possible, we focused on exploring the Openmoko capabilities to command a remote auralization session. It happens that current local memory is not large enough for holding all the sound files in the phone.

4.1 Perspectives and future work

An auralization system portable in a mobile phone is indeed a powerful resource for audio engineers and music professionals. New mobile phones bring touch-screen graphical interfaces and offer lots of communication layers, such as Bluetooth, 3G telephony and Wi-Fi networks, and can also play sounds and capture video. Not

only new applications may become possible with this, but also current audio production tasks can be spread onto a set of platforms such as the usual mixer consoles, gear racks and audio workstations, and also onto portable phones, performing for instance the control of a chain of processes. This, for example, plugs mobiles into the DJ's scenario and can stimulate the emergence of new collaborative music composing and performing in social nets, as well as collaborative soundscape building, among many other exciting possibilities.

The demand for cell phones and applications has been growing in the world for several years, as well as the demand for creative solutions and innovative technologies in multimedia. In this sense, the realization of a portable auralization engine increases the range of existing solutions for mobile audio computing, not only for phones, but also for handheld devices with larger storage and processing capabilities. However, challenges for a next generation of mobile phones include provision for multichannel audio output ports and spatial audio middleware for sound-based applications.

This working prototype is to create conditions for new initiatives for the application of spatial audio in portable platforms. Future works address naturally a second round of developments, improvements and tests, including concurrent audio applications, memory expansions, and Wi-Fi / 3G transport layers explorations. These wide-band transport channels are expected, for example, to make stand-alone auralization feasible by receiving streamed sound objects.

5 ACKNOWLEDGMENTS

We would like to thank Jon "Maddog" Hall and the Canadian-based company Koolu for fully supporting this initiative and providing one Openmoko unit for tests, and to Renan S. Vital, who devoted to Openmoko programming and contributed to the project.

REFERENCES

[1] Faria, R. R. A. and Zuffo, J. A. *An auralization engine adapting a 3D image source acoustic model to an Ambisonics coder for immersive virtual reality*. In Proceedings of the AES 28th International Conference (The Future of Audio Technology – Surround and Beyond, Piteå, Sweden, June 30-July 2, 2006). AES, New York, NY, 2006, p.157-166.

[2] Faria, R. R. A. et al. *AUDIENCE – Audio Immersion Experience by Computer Emulation: Systems for Audio Immersion, Production and Reproduction of Spatial Audio*. 2007. Available at http://www.lsi.usp.br/interativos/neac/audience, Accessed August 10, 2009.

[3] Faria, R. R. A. et al. *OpenAUDIENCE – Audio Immersion Software*. 2008. Available at http://openaudience.incubadora.fapesp.br/portal, Accessed December 10, 2009.

[4] OpenmokoWIKI *Main Page*. Available at http://wiki.openmoko.org/wiki/Main_Page/, Accessed December 10, 2009.

[5] Puckette, M. S. *Pure Data: another integrated computer music environment*. In Proceedings of the Second Intercollege Computer Music Concerts, Tachikawa, Japan, 1996, p.37-41

[6] Pure Data – *Pd community site*. Available at http://puredata.info, Accessed April 10, 2009.

[7] Thomaz, L. F. et al. *Orchestra spatialization using the AUDIENCE engine*. In Proceedings of the ICMC 2006 (International Computer Music Conference, New Orleans, USA, November 6-11, 2006).

[8] Cabral, M. C. et al. *An experience using X3D for virtual cultural heritage*. In Proceedings of the twelfth international conference on 3D web technology (Perugia, Italy, April 15-18, 2007). Web3D 2007. ACM Press, New York, NY, p.161-164.

[9] Openmoko, Inc. *GTA02 - Neo Freerunner Schematics*. *2008*. Available at http://downloads.openmoko.org/schematics/GTA02/, Accessed July 18, 2008.

[10] *QEMU open source processor emulator*. Available at http://www.nongnu.org/qemu/, Accessed August 21, 2008.

[11] Gnocl. *Tcl meets GTK+ and Gnome*. Available at http://www.dr-baum.net/gnocl/, Accessed June 1st, 2008.