

# An Open Platform for Multimedia Entertainment Systems

Marco Lohse and Philipp Slusallek  
Computer Graphics Lab, Department of Computer Science  
Saarland University, Saarbrücken, Germany  
{mlohse, slusallek}@cs.uni-sb.de

## Abstract

Today home entertainment appliances are usually based on closed, proprietary hardware and software design supporting only a limited set of media types, stream formats, and entertainment options. With the availability of cheap, small, quiet, and powerful multimedia PCs, it is now possible to create an open and extensible PC-based platform for multimedia home entertainment. Based on a new multimedia middleware for Linux, we describe the design of a simple, and extensible application and user interface framework. We have used this framework to create a home entertainment application that currently supports playing CDs, MP3s, DVDs, grabbing CDs, as well as watching digital and analog TV with time-shifting.

## 1 Introduction

Almost every household has one or more systems that provide multimedia home entertainment. While traditional systems like radio, TV, CD, MC, and LP are still widely used, there is a clear trend towards media convergence, where most of the home entertainment options will be based on digital media existing in various formats.

The market for home entertainment systems is fairly large and is currently dominated by inflexible, proprietary, and closed hard and software systems. On the other hand more and more households already own a PC that is fully capable of providing all the necessary multimedia services if equipped with suitable I/O devices, such as analog or digital TV tuner, DVD player, etc.

However, there is hardly any PC based software that provides the necessary services. Microsoft's MediaPlayer and Apple's Quicktime

system [8, 2] provide some basic functionality but do not offer an extensible application. On the other side is the Linux operating system, that is gaining a stronghold in the PC and embedded area, but is missing a suitable multimedia middleware and coherent applications on top of it.

In this paper we describe a flexible application and user interface framework for creating multimedia home entertainment systems: the *Multimedia-Box* [10]. The framework builds on top of the NMM multimedia middleware for Linux [9], which provides services roughly equivalent to DirectShow from Microsoft but adds network transparent data access and device control [6].

On top of this middleware we created a flexible framework for a multimedia applications. The core application is extensible through plugins that support different entertainment options and media types. We currently provide plug-ins for playing CDs, MP3s, DVDs, and analog and digital TV streams. The system also provides advanced features such as MP3-encoding with CDDDB ID3-tags [3] and time-shifting for TV.

A general but very thin user interface layer describes the appearance and the event handling. The application and the user interface are configured through simple XML configuration files. The applications functionality and its appearance at the user interface level may easily be modified simply through switching between different XML configuration files.

### 1.1 Related Work

Recent developments have shown, how an ITV system can mediate interpersonal communication with user tracking and communication services [1]. The Multimedia Home Platform (MHP) [12] specifies an interface between interactive digital applications and the terminals

they run on. As MHP only defines this interface, applications are needed as well as a middleware layer, like the OpenTV SDK [11].

In [5] the concept of a broadcasting consumer terminal is presented, which is based on an implementation of the Multimedia Home Platform (MHP) specification. This approach comes closest to our system, however we focus on providing a unified access to different media types, stream formats, and format conversions. Our system provides an advanced set of functionalities, like time-shifting and transcoding, and offers an open and extensible platform for further developments.

## 2 Hardware Setup

The design goal for the *Multimedia-Box* was to create an open and extensible home entertainment platform that runs on commodity PC hardware. For this reason we have chosen a Linux-based PC with multimedia extension boards over special purpose hardware, e.g. for settop boxes. For user presentation and interaction the PC is connected to a TV and loud speakers as in most living room scenarios. The Multimedia-Box is controllable with only a standard remote control even when the TV is switched off. To this end we integrated an infra-red receiver and a small LCD display. Since our solution should also provide the convenience of today's consumer devices, we chose a special chassis and low-noise components.

For receiving TV programs two options are supported: A DVB-board directly receives MPEG-2 encoded digital TV, while for analog TV we use a combination of a TV-tuner board and a cheap KFIR MPEG-encoder board.

## 3 Multimedia Middleware

The Multimedia-Box is based on the *Network-integrated Multimedia Architecture* (NMM) as its underlying middleware [9]. It is implemented in C++ and currently runs under Linux. It provides multimedia services similar to DirectShow but extends these services also to non-local devices located somewhere on the network.

Within NMM, all hardware devices (e.g. a DVD-ROM drive) and software components (e.g. decoders) are represented by so called *nodes*. Nodes have properties that include its

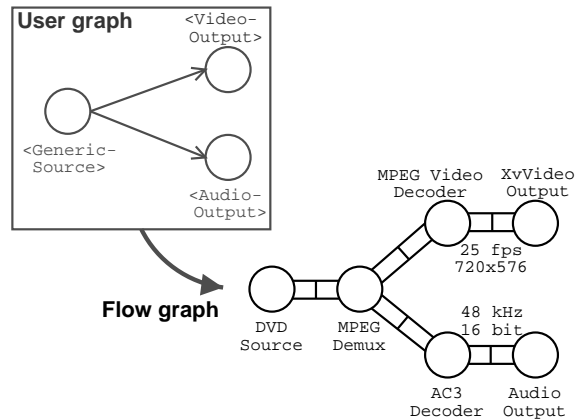


Figure 1: User graph and flow graph for DVD playback.

inputs and outputs ports, called *jacks*, together with their supported multimedia formats.

These nodes can be connected to create a flow graph, where every two connected nodes have to support the same format. The NMM framework also supports a quality-driven format negotiation. Starting from an incomplete flow graph (the *user graph*) that is independent of multimedia formats and needed converters (e.g. MPEG decoder), this procedure automatically creates a complete and fully configured flow graph by inserting necessary converters and setting formats [7]. Figure 1 shows an example of an incomplete user graph for DVD playback and a complete and fully configured flow graph.

The framework offers facilities for efficient memory management, scheduling, and a sophisticated state machine for all processing nodes. An unified event system and mechanisms for synchronizing different media streams with QoS control are also provided. Generic base classes exist that simplify the integration of new processing units and codecs by inheriting from a suitable class and only implementing the specific processing code.

Although the Multimedia-Box application is operating on a single machine, the NMM framework provides the infrastructure that allows to consider the network as an integral part by enabling support for devices distributed across a network [6].



Figure 2: The Multimedia-Box and the main menu rendered in different styles or *skins*.

## 4 User Interface

The user interface of the Multimedia-Box presents a set of hierarchical menus. At the leaves of this hierarchy are special “actions”, such as “DVD playback”. The user can navigate these menus with a remote control. Visual feedback is given via the connected TV and the integrated LCD display. Intuitive and descriptive icons with textual help, support the user in navigating the menu structure. The user interface elements also display feedback on the state of the application, such as during “play” or “pause”.

The main menu of the Multimedia-Box currently consists of four menu entries: Audio-CD, MP3, DVD, and TV. The “Audio-CD” menu offers two choices, “Play” and “Grab”, where the latter transcodes the audio data to MP3 and stores it on the internal hard disk. A request to a CDDB database [3], which can be locally installed or available on the internet, is performed to obtain the names of albums and tracks. The “MP3” menu offers access to the locally stored MP3 files. The “DVD” menu entry starts playing a DVD disc and offers access to the different menus, chapters, and titles. Finally, the “TV” menu entry directly switches to watching television, either digital or analog. This option is described in more detail in the next section. Figure 2 shows the Multimedia-Box and different graphical user interfaces.

## 5 Time-shifting

Besides switching of channels and starting recording of the currently viewed stream, our implementation also offers time-shifting of TV streams. This mode can be entered by pressing the “pause” button, which freezes the current channel. Instead it starts recording the stream to a circular buffer, first in memory and then — if a predefined memory limit is exceeded — on the local hard disc.

By pressing the “play” button, the program is continued from the previous position in the data stream, now playing from the buffered data. In this mode the user may fast forward to catch up with the original TV stream or seek backwards again. The system can also be configured to always store the most recently received data in a fixed size buffer. This allows to arbitrarily visit recent events in the stream, e.g. for a slow-motion review. Programming of desired recordings is currently not yet supported but will be added soon.

## 6 Architecture

A major idea in the design of the Multimedia-Box was easy configuration and modification of the user interface. In addition to easily changing of the look of the application with so called *skins*, this requires the ability to adapt and extend the structure of the application itself. Therefore the whole application is built as a plug-in-architecture that is assembled using a XML-based configuration language.

The XML configuration specifies a hierarchical menu structure: Each menu (e.g. “Audio-CD”) contains several entries that correspond to either sub-menus or to “actions”. Actions entries (e.g. “Play CD”) corresponds to methods exported by the Multimedia-Box application. Menu entries also require references to image files and their screen position for creating different screen layouts. These images represent an “active” and an “inactive” appearance. The structure of the menu itself can be controlled in terms of the number of rows and columns. Each menu entry can then be placed within this grid. Additionally, a background image can be specified for each menu.

From this XML-description the application is automatically set up: for each menu and action, a *MenuState* or an *ActionState* object is

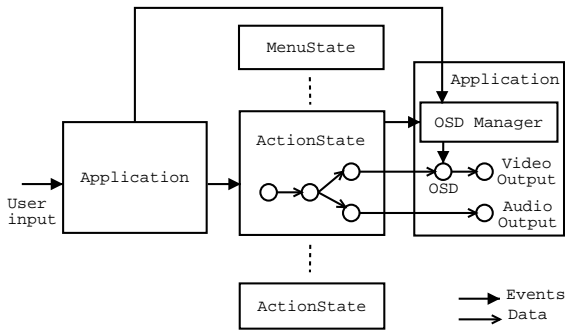


Figure 3: Architecture of the Multimedia-Box application.

created, respectively. Every *MenuState* object contains the graphical objects of its menu and is able to display the correct activated and deactivated items according to user navigation by updating and evaluating an internal state. As every *MenuState* object provides the same functionality it is implemented by a single class.

In contrast, the *ActionState* object offers application specific functionality that is provided by the application but might actually be loaded as a plug-in as requested by the XML configuration. Our current implementation provides four different *ActionState* objects: audio CD playback, MP3 encoding of an audio CD, MP3 playback including navigation in the local directory structure, DVD playback, and TV access with time-shifting. Internally, these *ActionState* objects use NMM nodes connected to flow graphs.

The overall system architecture consists of several interacting parts (see Figure 3). An event producer (user input), the main *Application* object, a number of automatically instantiated state objects (either for menus or actions), and an *OSDManager* object.

The event producer generates commands according to user input with a remote control. These commands are sent to the application object and are forwarded to the currently active state object. The state objects implement the *State* design pattern [4]. Depending on the received events the active state object performs an internal state transitions (e.g. from “play” to “pause”) or an external state transition by activating another state object.

The sink nodes for audio and video output are used by all states and are included in the application object. Additionally a node for blending on-screen elements onto the current background is inserted before the video sink node.

This node is controlled by the *OSDManager*, which provides a more high-level access and offers operations like “show symbol *pause* for 2 seconds”.

To obtain access to the audio and video output devices and the *OSDManager*, the currently activated state is connected to these nodes. When connected, the state object can render video and audio and change the elements of the on-screen display. Figure 3 shows the flow graph for DVD playback being connected. A *MenuState* object can use this facility to display an animated background and play sound effects as feedback for user navigation and selection events.

With this architecture, integrating a new action, for instance a video conferencing application, is straightforward. Implementing a new *ActionState* object identified by a unique string and adding a new menu entry in the XML configuration is all that needs to be done.

## 7 Conclusions

We have described an open and extensible home entertainment platform. It is based on the NMM multimedia middleware that provides reusable components that simplify the creation of advanced applications. Most of the effort in this project was spend in implementing the underlying NMM nodes (e.g. DVB card access, DVD library, etc.), which are now also available for use in other applications.

A flexible but fairly thin user interface layer supports arbitrary configurations of the applications through application plug-ins and XML-based configuration files. The XML files can easily be modified even by an end user to adjust the appearance of the Multimedia-Box application.

The Multimedia-Box application provided the perfect test bed driving the design and the implementation of the emerging NMM middleware. The Multimedia-Box project was originally initiated due to our personal desire to build our own PC-based home entertainment platform for our living rooms. We are now at the point, where we will be building several copies of our prototype system for real world testing at our homes.

Our system currently already supports the basic entertainment options. In addition we would like to see more options such as an elec-

tronic program guide (EPG), video conferencing, Internet telephony, chat, email, web access, and many others.

The NMM framework is already available as Open Source [9] and the Multimedia-Box application will be released soon. Hopefully this will encourage more people to improve and extend the current system and add many more media types, media formats, and entertainment options.

Additionally, we are extending the Multimedia-Box to access distributed devices controlled by the NMM framework. This will allow to make use of these devices as new data sources or to distribute time-consuming tasks like video transcoding to other devices.

## 8 Acknowledgements

We thank Marc Klein, Markus Sand, Wolfgang Enderlein, Patrick Becker and Patrick Cernko who implemented most of the underlying NMM nodes and the Multimedia-Box application. Michael Repplinger and Stephan Didas provided significant support on the NMM middleware level without which this project would not have been possible. This research has been supported by Motorola, Germany, and the Ministry of the Saarland.

## References

- [1] Jorge Abreu, Pedro Almeida, and Vasco Branco. 2BeOn - Interactive Television Supporting Interpersonal Communication. In *Multimedia 2001 - Proceedings of the Eurographics Workshop*. Springer Verlag Wien, 2001.
- [2] Apple. Quicktime 5 API Reference. <http://developer.apple.com/quicktime/>.
- [3] freedb.org. CDDB database. <http://www.freedb.org/>.
- [4] Erich Gamma, Richard Helm, and Ralph Johnson. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Massachusetts, 1995.
- [5] Klaus Ilgner and John Cosmas. System Concept for Interactive Broadcasting Consumer Terminals. International Broadcasting Convention, 2001.
- [6] Marco Lohse, Michael Repplinger, and Philipp Slusallek. An Open Middleware Architecture for Network-Integrated Multimedia. In *IDMS/PROMS'2002 Joint International Workshop on Interactive Distributed Multimedia Systems / Protocols for Multimedia Systems*, 2002, to appear.
- [7] Marco Lohse, Philipp Slusallek, and Patrick Wambach. Extended Format Definition and Quality-driven Format Negotiation in Multimedia Systems. In *Multimedia 2001 - Proceedings of the Eurographics Workshop*. Springer Verlag Wien, 2001.
- [8] Microsoft. Windows Media Player. <http://www.microsoft.com/windows/windowsmedia/>.
- [9] Network-Multimedia Workgroup. Network-Integrated Multimedia Architecture. <http://www.networkmultimedia.org>.
- [10] Network-Multimedia Workgroup. Linux Multimedia Box. <http://www.networkmultimedia.org/NMM/Status/MMBox/>.
- [11] OpenTV. Multimedia Home Platform (DVB-MHP). Technical White Paper, 2000.
- [12] Tom Worthington. Internet-TV Convergence with the Multimedia Home Platform. Communications Research Forum, 2001.