The Evolution of the Graphic Editing Environment for the IRCAM Musical Workstation

Enzo Maggi, François Déchelle

The Evolution of the Graphic Editing Environment for the IRCAM Musical Workstation

Enzo Maggi, François Dechelle

Proceedings of the International Computer Music Conference, Hong Kong (1996)

Abstract

Since its introduction, the 'Max' programming language and real-time environment has had a continue evolution: it was separated from its computational engine (the FTS system), it evolved into a full featured industrial product (Max IRCAM/OpCode) and experienced a constant growth of its external modules library. From the beginning, Max had an object-oriented approach to the description of computations (patches) and to the extension of the system through external modules. Now, the Max paradigm is evolving again into a system that is capable of dealing with complex and more rich types of intercommunications, such as the ability of being a "part" into a compound document, and at the same time to communicate with fast real-time engines such as FTS.

1. Introduction

Speaking about the graphic environment for the IRCAM worksation, after many years of its introduction, means to deal with a paradigm that imposed a standard in the Computer Music community. Max infact introduced for the first time in a real time system the concept of graph based execution, with a message-based logic very close to the modern object oriented paradigms.

The advantages of such an approach are much greater, from the user point of view, than the limitations imposed by the different implementations. This has been demonstrated by the wide success in the user community.

Max macintosh and Max workstation have two different set of functionalities and different targets. The first is suited mostly for MIDI data handling and therefore used in small MIDI-equipped studios. On the other side, the workstation implementation provides the user with real-time audio signal processing, using a well-defined real time engine (FTS) and a graphic interface that is a client of such an engine. The target of this environment is the production of computer music works and the research and experimentation in the field of signal processing.

This paper describes the main choices followed in the development of the environment together with a view at the current and future developments.

2. The Max / FTS paradigm

A first important hypothesis that is at the base of the new developoment of the environment is the possibility to separate the "Max graphic interface" from the "Max computational engine". This separation led to the possibility to focus the development toward the direction of a real-time DSP engine as described in the introduction. At the same time, this was the building block for a client-server architecture that made it possible to write client applications (other than Max) on the same computing base. Today, it is possible

with a little effort to control the synthesys functionalities of the IRCAM station by the mean of specialized editors.

The second step toward a more flexible architecture has been the choice to make the FTS engine as portable as possible on a wide spectrum of platforms. At the present, the code base of FTS is highly portable on the platforms that have standard ANSI-C compilers (UNIX, Macintosh...). This makes the underlying paradigm even more independent from the platform.

The introduction of a communication protocol between the client and the server made it possible to connect a client to an FTS server in a "remote" way. Max Ircam/OpCode itself can be one of the clients of FTS [Dechelle and Dececco, 1995].

3. The new directions

The rapid growth of performance on low-cost processors like the PowerPC allows the port of the FTS engine on the Macintosh platform [Dechelle *et al.*, 1996]. The target of such a process is to extend the available platforms for FTS, providing a large spectrum of users with the audio sysnthesys capabilities of the IRCAM workstation, while preserving at the same time the musical works made on different version or even on different platforms of the system.

The implementation of the system on a new platform brought the IRCAM engineering team to a redesign of the underlying architecture in such a way that it provides the user and the developer with enhanced new features. This can be achieved by designing a modular system based on portable layers of software, looking for a more general solution to the problem of integrating the computational engine with the graphic user interface.

The new architecture will be embeddable in the sense that the integration of the system in other complex systems will be as easy as possible; furthermore, the separation of different functional layers of software makes it possible to easily evolve the architecture as soon as other software technologies become available.

The different layers of software, that will be analized more in depth in the next paragraph, allows a modular design in which each 'layer' can become an active reusable part, for example in an OpenDoc context.

Of course, a strong accent was put on the portability of the whole system, in its computational part as well as in the graphic part. A portable file format specification will make it easier to exchange information between platforms while mantaining compatibility with the old patch file formats.

The best choice is the one that allows at the same time the integration of the various sub-systems and the evolution of their functionalities, preserving the user investments in terms of user-experience, patches and external objects.

4. The modular choice - the FTS Editor

The new architecture have a modular structure, a clear separation between graphic interface and computational engine and well-defined APIs for the interaction between the client and the two underlying servers. The choice of a reusable and portable graphic layer carries to a simplification of the client code, as shown in fig. 1

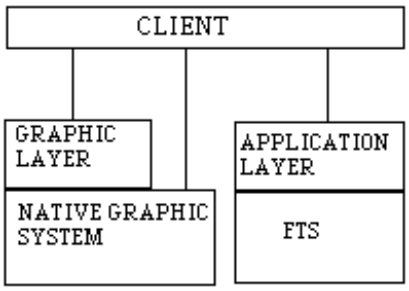

Figure 1: the software architecture

The portable graphic layer provide the user with the ability to create and handle simple graphic objects and simple data types (coordinates, sizes, fonts). A good strategy to achieve a certain degree of (client) portability is to define the interface that the graphic layer presents to the application, hiding the implementation that will be platform-dependent. Of course, a complete graphic application would use more capabilities than the simple abstraction provided in the graphic layer, so an effort will be required in order to port the parts of a client that are platform-dependent.

The application layer, on the other side, is a software component whose main aim is to provide an interface for client applications (not only max) providing them with a set of FTS-related services. These services also allows an application to communicate with FTS with a "high level" programming interface.

This layer is a mirror of FTS objects, defining the FTS object abstraction, and offering other services such as file I/O, direct-to-disk, and so on.

The graphic of the FTS Editor is built on top of the portable graphic layer, which is not a complete substitution of the native graphic on each platform. Even if portability graphic toolkit exist currently on the market, the development of software based on these toolkits poses a big limitation on software reusability, and force the software development to follow the toolkit life-cycle as a product.

The interactions between the client and its servers are based on registration-callback mechanisms, in such a way that each subsystem could be easily re-implemented with different technologies (ex. OpenDoc component) without big modifications of the client code.

5. Conclusions

The new strategy for the evolution of the graphic editing environment involves the FTS integration into new high-performance PowerPC native implementations, the design of a new editor based on portable layers of software, up to the specification of a common layer for the synchronization and the data exchange between the musical applications currently developed at IRCAM (OpenMusic architecture).

The dynamic linking facility of the new editor is going to be extended in order to communicate with non-local objects and services, with the future perspective of patches execution that is indipendent from the physical location of the components involved in the computation.

The project will have as immediate benefits the integration of audio capabilities, and the possibility to build compound documents in which 'Max like' patches will play the role of interconnected real time computation.

References

[Dechelle and Dececco 1995]
François Déchelle, Maurizio De Cecco. *The IRCAM Real-Time Platform and applications.* ICMC 95, Banff

[Dechelle et al., 1996]
François Déchelle, Maurizio De Cecco, Enzo Maggi, Norbert Schnell. *New DSP applications on FTS*. ICMC 96, Hong Kong.