Teaching, investigating, creating: MUSICOLL

Anne Sèdes, Alain Bonardi, Eliott Paris, Jean Millot, Pierre Guillot

CICM

Université Paris 8

2, rue de la liberté

F-93526 Saint-Denis Cedex

MSH Paris Nord

20 avenue Georges Sand

F-93210 Saint-Denis La Plaine

anne.sedes@univ-paris8.fr

INTRODUCTION

In the Department of Music at the University of Paris 8[1], innovative tools and methods to teach music and signal processing are at the centre of our concerns, as teachers, researchers, and creators. In the following collectively written text, after a focus on the academic context and our recent activities, we will present our new project, called MUSICOLL. It is centered on collaborative and portable real time music in the context of teaching and creating sound, music, computer and music sciences and methodologies.

1. Teaching, researching, creating

W e tackle the previously mentioned in the field of Music and Computer Science in a Music and Musicology programme, in

1 http://www.univ-paris8.fr

interaction with the University's policy for promoting science in the arts, science and technology, research-creation, and digital humanities.

As a minor specialization in standard Music and Musicology Bachelor of Arts (licence), Master's and Ph.D. (doctorat) degrees, the courses offered to our students concentrate on musical composition, an introduction to programming languages in relation with digital audio, and academic theoretical approaches linking research with creation. The process of playing? electroacoustic, live electronic and mixed music and their intermedial extensions, are at the core of our pedagogy, where all students make their own way through the arts, technology, science, and humanities to develop their own specialization while supervised by a team of more advanced students, Ph.D. students and research professors.

Mixed music is a precious field for experimentation, teaching, creation, research and software development. Within this area, thanks to software like Max[2] and Pure Data[3], a sound spatialization thematic devised by musicians and for musicians allowed us to elaborate the HOA[4] [Sèdes & al. 2014] project . It has been developed over the course of four years as a part of the LABEX Arts H2H[5] with joint development from the arts , software and software engineering fields. It gave rise to thesis projects and the completion of a set of libraries already well-known by the international community. The project is still in existence, continuing to develop and with the aim of enduring operating systems' and software environments' evolutions.

The HOA project had a relatively modest goal: a C++ library for high order ambisonics allowing for several software implementations like plugins, to develop graphical interfaces for the field of sound visualization, offering a 3D approach, embedding sound transformations within the spherical harmonics domain, etc.

HOA-led workshops allow Music students to create sound spaces, presenting questions of musical composition in programming for pieces produced in [6]the composition workshop with recognized professional ensembles such as the Percussions de Strasbourg, Ensemble Itinéraire, Ensemble Aleph...

Regarding software creation, doctoral students Julien Colafrancesco, Pierre Guillot, and Eliott Paris contributed the different versions and revisions of the project, under the direction of Anne Sedes and Alain Bonardi at CICM.

In an informal way, the methods were based on learning through hands-on experience, modelling, trial and error, testing and redesign, in interaction with direct feedback from users who were connected remotely and experienced in studio work?, composition, and dissemination.

The limitations of MAX, with its relatively closed-source proprietary code and the significant and unforeseen changes made by its developers in the newer versions, and Pure Data, in open source, but always requiring serious revisions [Guillot 2014], led the team to consider new approaches in an environment where collaboration and portability or the cloud are now becoming strong trends. Thus, the MUSICOLL project was launched. Its purpose: to renew real-time music practices, and move towards collaborative, multi-platform, perennial patch creations.

The objectives are :

1. to produce a framework of a new collaborative and mobile real-time processing software

2. to study the handling of this software by creators

3. to examine how to renew real-time music software teaching

4. scientific and professional dissemination

In the following, we will present the status of the project regarding its software developments, over the course of ten months.

2. MUSICOLL: the project and its first results after ten months

Collaboration between geographically displaced musicians through Internet networks has become more and more popular, the origins of which can be found in performing and playing music together [Renaud & al. 2007]. The next step consisted of getting several people to shape audio contents together, for instance using a collaborative sequencer like OhmStudio developed by OhmForce[7]. We now

consider the collaborative design of real time sound processors possible due to a graphic language.

The MUSICOLL project is an ANR-funded project that unites the CICM, which is located in the Maison des Sciences de l'Homme Paris Nord, and a company, OhmForce, specialized in collaborative digital audio. It aims at redesigning the musical practice of real time, which is becoming more cooperative and mobile. The project is expected to last three years, from January 2016 to December 2018, and proposes the development of the first draft of a musical real-time collaborative environment enabling several creators to work simultaneously on a process hosted online accessible from any connected terminal. This environment for the graphic programming is called Kiwi. We will also study how music creators become familiar with Kiwi and use it. Kiwi will then be presented in class, for students that are beginners in real-time processing, and observe the resulting educational renewal. Last, but not least, we will undertake actions to disseminate the results to the Computer Music community and to sound professionals.

The first expected result is the production of the prototype of a Kiwi application providing an underlying kernel of sound transformations and syntheses to respond to a first level of composers' expectations. In its first version to be released at the end of October 2016, Kiwi will be a network application running on Macintosh computers, based on a collaborative point of view on a Flip framework that is developed by OhmForce. It is described in the third part of the present article. At the end of the project, we should achieve a plugin version of the application that could be integrated into digital audio workstations like OhmStudio, enabling to embed collaboratively designed real-time processes in a sequencer.

Concurrently with Kiwi's software development, two usage studies will commence in 2017, directed towards two significant communities at The University of Paris 8: composers on one side and inexperienced students in real-time sound processing on the other.

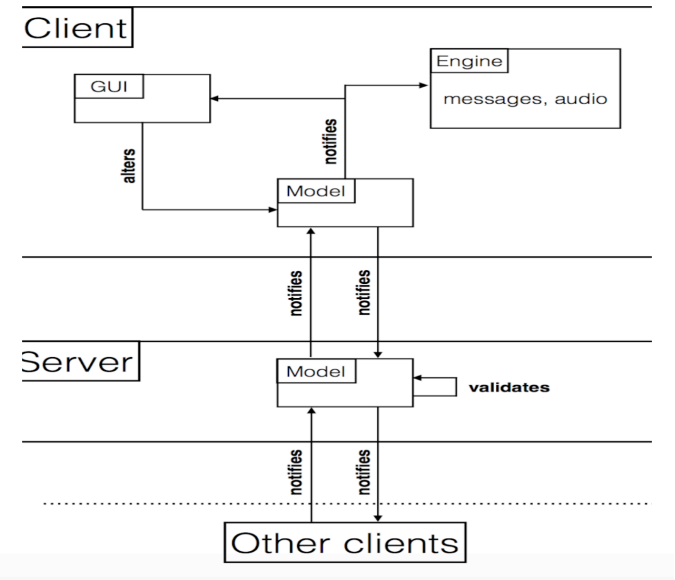


Figure 1. *Diagram of the current architecture of the Kiwi application.*

This research project raises several issues: the first of which are technical, dealing with signal processing as well as synchronization through networks, but also design problems in collaborative approaches, and finally those of appropriation and diffusion, trying to encourage people to work differently, using their feedback.

3. The Kiwi application: the architecture of the prototype

Kiwi is an application that allows one to create real-time audio effects by connecting graphical boxes called "objects" inside a graph named "patch". This approach, similar to the one offered by the Max and Pure Data software, takes another dimension due to its collaborative and nomadic aspects. Indeed, this project allows users to design and create audio engines together by sharing and testing their ideas in real-time in the same "patch" and in a common workflow. The collaborative features have been made available by OhmForce through Flip[9], a C++ framework for creating collaborative applications in a model-view-controller (MVC) architectural pattern[10], and managing transactions over the network. The graphical and audio rendering is

carried out by JUCE[11], a cross-platform framework commonly used in the audio application industry.

The software is composed of two distinct parts, one for the client and the other for the server. The client part is destined for the user and there can be as many versions of it as there are users. The server is unique and customers can communicate through it. This communication is possible thanks to a common model shared by these two parts, the client and the server. This model based on Flip represents all the persistent and collaborative data of a patch. The patch is mainly composed of a set of objects and a set of links. An object is represented by a name, a certain number of inlets and outlets. It is important to note that the object's functional part, its processor, does not belong to the model. The model is only an abstract description of the data managed by the application. At last, a link is represented by two objects' references and the indices of the inlet and the outlet connected.

When the client part modifies its model (by changing a link or adding an object to a patch for example), this model automatically sends a transaction to the server that dispatches the modification to all the other clients. The server part can also invalidate a modification if, for concurrency reasons, this alteration cannot be applied (for example, if a link has been created by a user and another user deleted one of the objects bound to this link at the same time). In this case, the server can choose to restore the user data model to a previous valid state and oblige all users to apply the same modification (the object will be deleted, and the link will not be created for example).

As has been previously suggested, the client part uses the model and manages other components: the graphical user interface (GUI) and the engine. The GUI part is composed of a view and a controller. The view part of the GUI graphically renders the model by displaying the patch, the objects and the links to the screen. When the patcher model changes, the view receives a notification and modifies its rendering. The controller part receives user interactions, coming from the mouse or the keyboard for example, and translates them into model modifications. Therefore, in the MVC architectural pattern, the GUI part is both a view and a controller. On the other hand, the engine is the computing kernel of the application. The fact that the engine is a

view concerning an MVC event even though it does not embed any graphical functionality can be misleading. As explained before, the model is autonomous and not directly linked to the engine. Thus, as the view, the engine receives the notifications when the model changes and interprets it. The engine owns the processors (or the brain part) of the objects. For example, an object called "+" in the model is in fact a name but the engine interprets it as the addition operation. When a link is created in the model, the engine will bind these two objects, enabling them to communicate through messages.

There are two kinds of operations an object can process. The messages are the first ones that can result from mouse and keyboard interactions but also MIDI events or OSC messages. The computation of these operations happens at a relatively low rate. The other operations are performed on digital audio signals at a high sampling rate (between 44100 and 196000 Hz). As the model doesn't have any computational intelligence, the engine owns this functionality by checking and sorting operations, optimizing processes and managing operations' relationships between messages and audio signals. Therefore, the engine creates effects, synthesizers, and even more complex sounds by linking processors and transcribing messages into mathematical operations.

Thus, the application possesses a multi-layer architecture with intricate connections between the different layers which enable it to manage the many implications produced by the collaborative and nomadic aspects of the project. These aspects will be discussed in the next section.

4. The implications of the collaborative and nomadic aspects

The current release of the Kiwi application functions on a local server with one computer that hosts the patch and several users that can modify it. For the moment, the application only offers very basic interactions: creation, destruction, and shifting of graphical objects and chords. These modifications to the model are managed by a Flip framework. Thus once the server has taken them into account, it automatically notifies in return all the listening machines of the changes. Nevertheless, we are just beginning to manage these

interactions, and have already encountered some concurrency problems. Indeed, what should happen if someone deletes an object while someone else connects it to another object? Should the object survive and the link be created? Or does the link have to be ignored and the object deleted? Indeed, one of these interactions should be ignored; both cannot coexist. For the moment all these concurrency issues remain unresolved; anticipating and foreseeing the problems will be one of the project's main tasks. To respond to these issues, we plan to define use cases, to offer a set of solutions and evaluate them utilizing user feedback (the organization of the tests is presented in the next section of this paper). Ultimately, we hope to define a priority system where each action is hierarchized regarding the others. For example, if the deletion of an object has a higher priority than the shifting of an object, deleting an object while another user displaces it will result in its removal.

Beyond the concurrent model's modification, other concepts are directly linked to a collaborative workflow. The way user interfaces are made has to be revisited in collaborative environments, finding a way to display other users' interactions enabling them to quickly apprehend what is happening without increasing the information entropy of the patch is key.

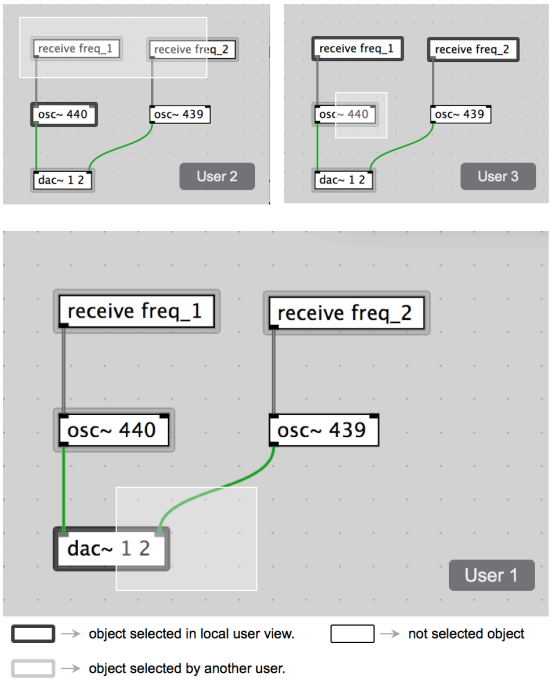


Figure 2. *This capture represents three users' selections interacting in real-time in the same Kiwi patch.*

User-1 can identify his/her own selection by its dark outline, but can also visualize other users' selections which are outlined in grey.

An example of these interactions is the selection of an object. Currently, the assumption is that knowing if one or more other users is selecting an object at the same time is more important than knowing who is selecting which object. The OhmForce team advised us to define a scheme that highlights the outline of the object using different colors based on the following rules: one color is assigned to the local user selection, and another is used to show that an object has also been selected by someone else on the network [see Figure 2]. One of this method's advantages is that it uses only two colors, allowing other colors to be used for displaying different information in the patch. It also improves the patch readability. Nevertheless, this information is partial. One might want to know which user selects which objects like in the Google Docs application[12]. This can be realized by using a set of distinct colors where each user is assigned one and their selections are customized using it, but this approach can make reading a patch

difficult. Another idea would be to have a window appear with information about the users making the selection as the mouse hovers over the objects. This approach which seems satisfactory on the surface is, in fact, a problem for tactile interfaces. Therefore, we will once again make suggestions to a set of testers to find the solutions that best match the needs of the Kiwi application.

This object selection feature is one among many caused by the collaborative aspect of a patch edition. For example, the OhmForce team's experience in the collaborative domain introduced an issue that we would never have imagined: if a user deletes an object or moves the object beyond the boundaries of the patch's window, these two interactions will appear to the other users as the disappearance of the object and none of them will be able to determine quickly what actually occurred. In the OhmStudio application where there were similar cases, OhmForce solved the issue by creating animations on the graphical user interface to notify passive users of this kind of interaction (by sliding or fading the GUI).

These problems were troublesome, but some of them were so significant they made the the application unusable. For example, often the user wants time to test and listen to the intermediary audio results in the patch creation process. The problem is that it can take a long time, and other users may want to modify the patch during their cocreator's listening session; but any modification of the patch will most likely result in audio glitches and artefacts. One solution we considered but which remains to be proved is offering an option that locks the patch and disables its updating while the audio is on. After the listening, the patch would retrieve the information and be synchronized again.

Thus, many issues can only be discovered while using and testing the first releases of the application, experimenting with both the collaborative and the portability aspects which also lead to many issues needing resolution.

5. Use cases, elements of demonstration and validation

At present, we have started working on usages, with a first scenario for demonstration and validation. This scenario was set in collaboration with OhmForce, and stage two users, each of them working with Kiwi on a common patch on his/her computer. This

scenario simulates a short workshop where user A explains to user B how to handle a simple oscillator and then several oscillators by duplication. It provides situations where collaboration happens either simultaneously or successively and highlights many collaborative issues specific to real-time graphic environments.

The Kiwi real-time environment we are developing requires validation through different types of uses and by specific communities interested in particular applications. A first example is teaching graphic languages such as Max or Pure Data to beginners, and in our case, undergraduate students in the Music Department of The University of Paris 8[13]. Its purpose will be to design and test the founding principles of a course based on the Kiwi collaborative platform, by comparing previous and newer educational practices to introduce real-time audio programming. A Kiwi trial class shall also be introduced. Teaching and learning practices will be documented with video recordings. Students and teachers will also be interviewed, but the documentation method for their responses has yet to be determined.

CONCLUSION

The MUSICOLL project and its Kiwi software have only been in existence for ten months. Therefore, there is much work remaining, especially concerning development. We currently have a small number of objects available in Kiwi, and more will need to be added to allow composers and students to test the application in a meaningful way. As the prototype is currently only working in a local area network, one of the next steps will be to make it accessible online, allowing users to save their documents directly online and join the same patcher document even when not located in the same place. In order to be effective, these developments will have to take place in close interaction with beta-tests to either validate or invalidate original specifications and react to users' feedback.

Over and beyond MUSICOLL, we have tried to demonstrate how a team working, teaching, researching and creating within a Music Department in a university can generate a favorable environment for obtaining new creation tools and providing skills for students. It can be accomplished via a software project based on informal and experimental methodologies in complete autonomy, and outside of the traditional industrial and engineering factoring models. We hope to

share it with potential communities adapted to teaching, investigating and creating, in order to utilize their feedback in our work.

BIBLIOGRAPHY

Guillot, P., 2014 : « Une nouvelle approche des objets graphiques et interfaces utilisateurs dans Pure Data » JIM2014, Bourges.

Jones, A., Kendira, A., Lenne, D., Gidel, T., Moulin, C., « The TATIN-PIC project: A multi-modal collaborative work environment for preliminary design », Proceedings of the Computer Supported Cooperative Work in Design (CSCWD) 2011 Conference, pp. 154-161.

Nam, T-J, Wright, D., « The Development and evaluation of Syco3D: a real-time collaborative 3D CAD system », Design Studies, vol. 22, No. 6, November 2001.

Renaud, A. B., Carôt, A., Rebelo, P., «Networked music performance : State of the art», AES 30th International Conference, Saariselkä, Finland, 2007 March 15–17.

Ruthmann, S. A., 2007 : « Strategies for Supporting Music Learning Through Online Collaborative Technologies », Music Education with Digital Technology, John Finney et Pamela Burnard (eds), London, Continuum International Publishing Group Ltd.

Sèdes, A., Guillot P., Paris, E., 2014 :« The HOA library, review and prospect », ICMC-SMS2014, Athens, Greece.

Schober, F. M., 2006 : « Virtual environments for creative work in collaborative music-making », *Virtual Reality*, pp. 1085-94.

Salinas, E. L., 2002 : « Collaboration in multi-modal virtual worlds: comparing touch, text, voice and video », *The social life of avatars* (pp. 172-187). Springer London.

Wilson, P., 1991 : « Computer Supported Co-operated Work: An Introduction ». Intellect Books, Oxford.

Zacklad, M., Lewkowicz, M., Boujut J.F., Darses F., Détienne, F., « Formes et gestion des annotations numériques collectives en ingénierie collaborative », Actes de la conférence Ingénierie des Connaissances 2003, pp. 207-224.

1 http://www.univ-paris8.fr
2 http://cycling74.com/products/max
3 http://puredata.info/
4 http://www.mshparisnord.fr/hoalibrary/en
5 http://www.labex-arts-h2h.fr/?lang=en
6 CICM stands for "Centre de recherche en Informatique et Création Musicale" meaning "The Centre for Research in Computer Science and Musical Creation". http://cicm.mshparisnord.org/ CICM belongs to the MUSIDANSE TEAM EA 1572 of the University of Paris 8.
7 http://www.ohmstudio.com/
8 ANR stands for "Agence Nationale de la Recherche" which is the French governmental agency for research.
9 http://irisate.com
10 http://martinfowler.com/eaaDev/uiArchs.html
11 https://www.juce.com/
12 https://www.google.com/intl/us/docs/about/
13 Approximately 20 students attend this kind of course every year.