Integra Live: a new graphical user interface for live electronic music

Jamie Bullock
Birmingham Conservatoire
Birmingham, UK
jamie.bullock@bcu.ac.uk

Daniel Beattie
Beelion Interactive
London, UK
dnl.bttie@gmail.com

Jerome Turner
User-lab, BIAD
Birmingham, UK
jerome.turner@bcu.ac.uk

ABSTRACT

In this paper we describe a new application, Integra Live, designed to address the problems associated with software usability in live electronic music. We begin by outlining the primary usability and user-experience issues relating to the predominance of graphical dataflow languages for the composition and performance of live electronics. We then discuss the specific development methodologies chosen to address these issues, and illustrate how adopting a user-centred approach has resulted in a more usable and humane interface design. The main components and workflows of the user interface are discussed, giving a rationale for key design decisions. User testing processes and results are presented. Finally, a critical evaluation application usability is given based on user-testing processes, with key findings presented for future consideration.

Keywords

software, live electronics, usability, user experience

1. INTRODUCTION

In this paper we present Integra Live, a new software application designed to address issues of software usability in live electronic music[1]. As musical practitioners working in a range of contexts including university-teaching, contemporary classical music and free improvisation, we have observed that existing software consistently presents an 'entry barrier' to musicians wishing to work with live electronics[2]. The most commonly used software for live electronics in an academic or 'contemporary classical' context is Max by Cycling 74[2][14]. Max was conceived as a 'graphical programming environment for developing real-time musical applications'[12], and as such it consists of a graphical dataflow language providing control data processing functionality for patchable digital signal processing (DSP) 'objects'. Max requires live electronics musicians to have an understanding of programming concepts such as conditional evaluation, iteration, mathematical and logical operators as well as DSP principles such as oscillation, filter design, delay buffering, table lookup and Fourier analysis. All of this is literally 'another language' to musicians who have devoted their lives to acoustic instrumental study or composition and simply want to experiment with live electronics. As a tool for dataflow programming and DSP, Max may be highly usable, but for musicians with little experience in this area, Max presents an unreasonably steep learning curve.

A number of existing projects seek to address this problem. For example, the Jamoma project[3] provdes 'a system for developing high-level modules in the Max/MSP/Jitter environment'[9], and more recently, a set of frameworks for developing Jamoma modules outside of Max[10, 11]. Jamoma offers significant advantages for both users and developers, presenting itself as a complete 'platform' within which processing modules may be used and/or developed.

An earlier project, Open Sound World (OSW), also sought to address usability issues identified in Max, by developing a new software application informed by user testing and usability evaluation[4]. However, like Max, OSW presented itself as a 'a scalable, extensible object-oriented language', and so, clearly targeted programming-savvy users rather than non-technical musicians.

Commercial software such as Bidule, Audiomulch, Reaktor, Ableton Live and Mainstage all have varying degrees of ease-of-use and applicability in live electronic music, with Bidule and Ableton Live being particularly popular with free improvisors and live electronic dance musicians respectively. However, due to its wide acceptance within academic institutions and research centres, Max remains the standard tool and entry route for composers working with electronics.

2. REQUIREMENTS

In order to verify our hypothesis that there is a need for a new application for live electronic music which is powerful yet usable for 'non-technical' musicians, we conducted a software requirements analysis. The purpose of this is to elicit requirements from stakeholders and potential users, and to analyse recorded data in order to establish design criteria.

2.1 Interviews

Four stakeholders consisting of: performer, professional composer, undergraduate composer, and post-graduate composer were interviewed in an informal setting. Interviewees were asked about their experience with existing software and informed about the aims of Integra Live and given the opportunity to respond freely about this. Some of the most salient comments are listed below.

> "I would like to see a piece of software that is more closely aligned to musical thought processes"
>
> "Current software is a big barrier for me using live electronics. It's a big deal for me to create the processing I need in my piece using Max"

[1] Music based on live processing of audio in performance
[2] `http://www.cycling74.com`
[3] `http://jamoma.org`

NIME'11, 30 May–1 June 2011, Oslo, Norway.

> "The basic processing modules should be already done so a composer can come and think about high level things. It would be nice if the most common processors were already there to be dragged or selected"

> "Max-like environments remove the element of play that you get with things like guitar pedals. These make more sense to the performer"

2.2 Online Survey

A survey of 76 potential users was conducted, drawing on Conservatoire students and staff, composers, members of new music ensembles, and members of Sonic Arts Network, Digital Music Research Network, British Computer Music and Canadian Electroacoustic Community mailing lists. 95% of those who completed the survey considered themselves to be composers with 68% considering themselves to be performers, and in general the results indicate that demographic group felt comfortable in at least two different roles. Most respondents reported that they used software for 'creation of new works', 'live performance' and 'experimentation', although 50% of respondents also use software for 'rehearsal', 'teaching' and 'writing new software components'. 78% of respondents indicated that they use 'live processing' software, with 85% indicating that they use software for 'Experimenting with sounds, controls, processing' and 76% indicating that they use software for 'performing live'.

Max was shown as being the most popular piece of software, with 21% of respondents indicating it as their favourite. SuperCollider[4] had 8% indicating it as their favourite with, Ableton Live accounting for 5%. Audiomulch and Bidule were both mentioned twice by those who indicated 'other' as their favourite software.

In addition to quantitative data gathered, the survey also recorded qualitative responses including reasons for liking or disliking specific software, and answers to the question: 'What features would you like to see in your ideal piece of music software?'. Salient responses include some of the following examples:

> "Everything. All in one. Allowing simplicity to complexity. For instance, most of people are using the basic function of Ableton Live but when you dig you can do really fancy things, programming kind of."

> "MUSICALITY. It has to work as a musical 'tool' not just as a software tool"

> Something like Max/MSP with an interface that's already in place, but that allow for infinite modification. Actually, I am thinking about AudioMulch for Mac, with a multitrack recording set up.

> An ideal piece of software would be able to be adapt to the users thought processes; this would make the software more intuitive to the user's own sense of logic. Sadly, every piece of software on the market requires a lot time just learning the program; having music software that was more 'human' would undoubtedly encourage more composers to explore different creative outlets.

> - easy way to connect any external hardware/instrument
> - few clicks to do tasks - uncomplicated first page
> - palettes of resources to choose from e.g. audio file pool, module pool

Common keywords include the following or their synonyms (number of people using the word out of the 64 who answered the section shown in brackets):

- easy (11)
- simple (4)
- like (15)
- user (9)
- interface (7)
- flexible (9)
- control (8)

The word 'like' was mostly used the context of a 'like X but with Y' idiom to indicate similarity to another piece of software e.g. '*Like* Main Stage but adapted for live electronics, simply, a piece of software like those but with flexible control over time, in concert and in rehearsal.'

2.3 ixi survey

In addition to our own surveys, conversations and interviews, we also drew on a recent survey conducted by the 'ixi' project[7] as part of our requirements gathering. This survey covers a slightly different demographic to the Integra survey, having a slightly greater emphasis towards participants with significant technical experience. This is reflected in the number of respondents reporting experience of tools that require some programming knowledge. Out of 209 survey participants, 52% indicated that they used Max/MSP, 49% indicating Pure Data and 40% indicating that SuperCollider. Interestingly, across all of the software indicated in the survey, the number of people indicating a program as their 'tool of choice' was relatively low compared to the number of users. For example, out of the 108 people that had used Max/MSP, only 35 indicated it as their tool of choice, and out of the 93 that used Reaktor, only 20 indicated it as their tool of choice. Across all of the applications in the survey the average number of users indicating a given application as their tool of choice was 17% of the total for that tool, suggesting a high level of dissatisfaction with available tools.

However, overall [7] suggests several classes of user, only some of whom are dissatisfied with available software. Particularly relevant to Integra Live are these findings:

> 'Some survey participants expressed the wish for more limited expressive software instruments, i.e. not a software that tries to do it all but "does one thing well and not one hundred things badly". They would like to see software that has an easy learning curve but incorporates deep potential for further explorations, in order not to become bored with the instrument. True to form, the people asking for such software tools had a relatively long history as instrumentalists.'[7]

2.4 Objectives

The survey results obtained, along with findings from interviews and informal conversations has led to the following observations:

1. A significant number of musicians from both acoustic and electronic music backgrounds feel that existing software for live electronics doesn't meet their requirements

[4] `http://supercollider.sourceforge.net/`

2. Many users currently use Max, so any alternative software must provide equivalent functionality, but with a musician-centred interface
3. Users require ease-of-control including the ability to easily connect external control sources
4. Users require a clean, well designed user interface that is somehow aligned with 'musical' thinking

Additionally, we aim to adhere to the following principles:

1. The software should behave like a normal application (i.e. *not* a framework or a programming environment)
2. The software should make the most common tasks easiest to achieve
3. The software should be visually appealing, and the visual design should enhance usability
4. The software should Just Work, providing low latency and stability
5. The software should be easy to download and install (good user experience)
6. The user interface should be self-explanatory [5]
7. The interface should favour standardisation over configurability
8. The software should hide complex functionality with simple UI

3. METHODOLOGY

As this project was part-funded by Integra (cf. section 8), the final application was required to be completed in a 12-month time frame. The project developers were divided between five research centres involved in the project: Birmingham Conservatoire (210 days), IEM (180 days), Notam (180 days), Malmö Academy of Music (45 days) and Muzyka Centrum (45 days). Each research centre agreed to a specific area of responsibility as follows:

- Birmingham Conservatoire: project management and core application development
- IEM: DSP module development
- Notam: Scripting functionality and Faust support
- Muzyka Centrum: module control development
- Malmö Academy: file format development and online storage

In order to deliver a user-centred application in the time frame given, a professional design company was contracted to collaborate on initial wireframes and graphic designs based on our detailed user requirements specification. This was then used as basis for an iterative development process. The Adobe AIR runtime environment was chosen as a platform for the GUI. This was due to its combination of portability, graphical richness and potential for rapid development. The Pure Data software was used as a DSP host partly for its parity with Max, but also for rapid DSP module development. Finally, the Integra Framework was developed as a middleware layer alongside the GUI to provide basic functionality such as file save/load, module management, host communications and an OSC interface[3].

Due to the disparate nature of the development team it wasn't possible to follow one specific development methodology, however, the general principles of the Agile manifesto[1] were adhered to:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Additionally, testing (user and functional) was given high priority, with testing results feeding into each iteration. This enabled us to minimise risk by finding problems early in the project.

4. APPLICATION WORKFLOW

Integra Live is divided into two main views, 'Arrange view', which is designed for arranging musical interactions in time, and 'Live view', which is designed to greatly simplify on-screen control during live performance.

4.1 Arrange View

The information architecture of the application follows a tree-like structure with a *project* at the top level. *Projects* may contain many tracks each of which may contain many blocks, each containing many modules. This is shown concisely in figure 1.

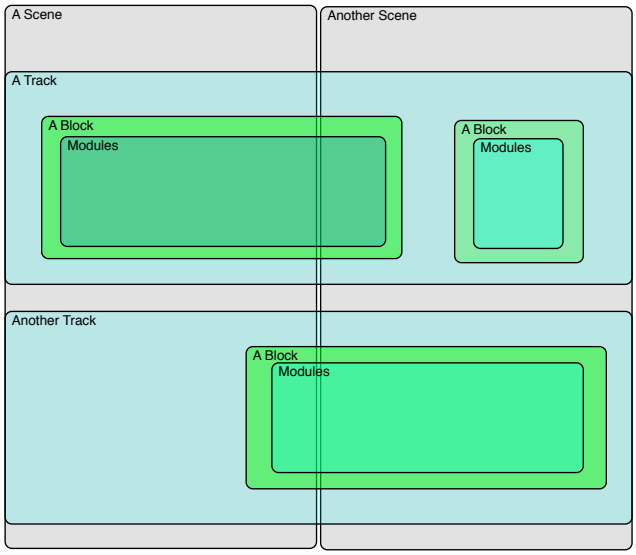

Figure 1: Integra Live information architecture

The arrange view represents this architecture visually, showing tracks as pairs of horizontal lines between which blocks can be placed. The user can navigate 'inside' blocks by clicking a '+' icon to expand them.

4.1.1 Module View

When the user navigates inside a given block, they are presented with a library of *modules* that can be dragged onto the block's canvas. *Modules* are discrete pieces of audio signal processing, synthesis or analysis functionality. It is part of the application's design that individual modules should be musically useful. That is, unlike software such as Max, which provides objects as 'building blocks' or primitives that can be combined to make more advanced units, Integra Live modules are aimed at immediate musical use. Another difference is that all Integra Live modules have pre-defined controls associated with with their attributes. These controls are displayed in the *module properties* window when a module is selected. The Integra Live core modules include a

range of filters, delays, reverbs, pitch shifter, granular synthesis, phase vocoding, resonators, soundfile playback and spatialisation. Module controls include slider, knob, range slider, x-y 'scratchpad', toggle, button as well as more specialised module-specific controls. In order to keep the user experience (UX) consistent, it was decided that the control type for each module attribute should stay fixed. That is, by design, users can't change the controls that are assigned to attributes. Figure 2 shows the module properties panel with the controls for the selected module.

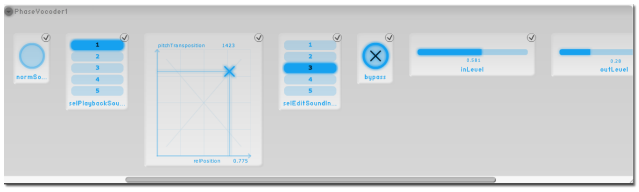

Figure 2: Module properties panel containing attribute controls

Module controls (such as sliders and dials) can be used to change *Module attributes* in real-time and are interactive through clicking, dragging and text-entry. Precise values can be entered by double-clicking the control's numeric value and typing the new value in the text-entry box (figure 3).

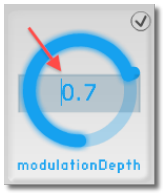

Figure 3: Entry of precise values into controls

The intended workflow in *module view* is that users create small groupings of interconnected high level modules and encapsulate these into *blocks*. The graphical presentation of modules is therefore relatively large to discourage the creation of highly complex networks of modules. If the user finds they can't achieve their goals without creating complex blocks, this would indicate a requirement for new modules providing the required functionality.

4.2 Live View

All modules and controls have a checkbox, which enables controls to be added to *Live view* either individually or per-module. The *live view* of the application shows all checked controls so that they can visualised and operated easily in live performance. Controls can additionally be resized and moved in live view. This design acknowledges that a different interaction model is required in live performance, where simplicity and immediacy of control are favoured.

4.3 Timeline

Integra Live has one master timeline, which is shown in the UI as a numbered horizontal strip near the top of the window. This provides a spatio-temporal reference point against which musical ideas can be organised. Timeline progression can be linear, where the ordering and duration of blocks corresponds to their ordering in performance, or non-linear, where the playhead moves to arbitrary points on the timeline with some blocks being activated indefinitely or stopped and started through user interaction. The playhead position can be changed manually by click-dragging the control triangle. The playhead state can be set to 'play' or 'pause' using the button controls in the top-left of the arrange view. Clicking the timeline numbering, can be used to zoom and scroll; clicking and dragging left/right scrolls, dragging up/down zooms.

Figure 4: Integra Live timeline and playhead

4.4 Envelopes

Envelopes provide a means to automate the control of module attributes over time. Envelopes are created by drawing control points into blocks in arrange view. Presenting envelopes in this way allows the user to visualise the musical 'shape' of a piece by looking at the arrangement of blocks and envelopes within the arrange view. Envelopes can be used for simple state changes (on/off) and for creating predefined musical gestures resulting from multiple module attributes changing simultaneously.

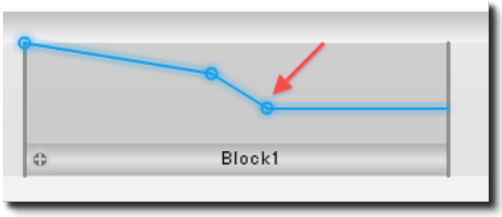

Figure 5: Adding control points to a block to create envelopes

4.5 Scenes

Scenes are used to create user-defined progressions through musical time. One application of this is to define Scenes that correspond to different sections of a musical work 'Section A', 'Section B', 'Cadenza' etc. Another application is to create multiple pathways within a work as found in improvisation and 'open form' composition. Scenes provide an additional layer on top of tracks and blocks, so that the playhead can be automatically moved to a given location on the timeline and optionally set to a 'play' state. This is achieved using *scenes*, which can be created by click-dragging in the space below the timeline. Like blocks, *scenes* have a duration, and additionally three possible states: hold, play and loop. If a scene is set to 'hold', when it is selected, the playhead will remain at the beginning of the scene. This means that all blocks under the playhead will become active and no further action will be performed unless the user gives further input to the system. If the scene is set to 'play' or 'loop' when selected, the playhead will proceed from the beginning of the scene and stop at the end of the scene or loop respectively.

4.6 Properties

When the software is in *arrange view* properties panels can be activated in the lower part of the screen by selecting entities within the UI. The properties panels follow a consistent layout, showing 'routing' and 'scripting' tabs for the

selected entity. The 'routing' tab is used to make connections between module attributes thus *routing* control data from one attribute to another. Multiple connections can be used to create one-to-many or many-to-one relations. Typical applications are for connecting external controllers such as fader boxes to DSP modules, and for 'ganging' attributes to be controlled in parallel. Activation of properties panels is shown in figure 6.

Figure 6: Block, Track and Project properties selection in arrange view

Clicking the *scripting* tab in the properties panel, provides access to the Integra Live scripting language. This is simple scripting functionality built on top of the Lua programming language. Module attribute state can be accessed from Integra script, allowing for procedural operations such as conditional evaluation and looping. The example below shows how the delayTime attribute of a TapDelay module can be modulated by side-chaining the audio input level.

```
x = AudioIn1.vu1
TapDelay1.delayTime = 100 / (x + 100)
```

5. USABILITY CONSIDERATIONS

Various features have been added across the application to improve usability and user experience. All entity instances are rename-able, allowing the user to add meaningful semantic information. For example, *scenes* can be given meaningful names like 'sectionA', 'coda', 'introduction' or 'improvsection'. Likewise, *tracks*, *blocks* and *module instances* can also be renamed.

All of these entities can also be exported from a given project and imported under a different node in the same project or into a different project. For example, the software allows a *block* to be renamed 'SpacialGranularSynth', and then exported as an Integra file for potential re-import. *Blocks* can additionally added to the in-application block library through a context menu that appears by context-clicking the *block*.

We stated in section 2.4 that Integra Live should behave like a 'normal' application. This means that it should meet the user's expectations on supported platforms, complying to the platform's human interface guidelines (HIG) as appropriate. Integra Live therefore has automatic association with its supported file type (.ixd). When a given .ixd file is double-clicked or dropped onto the application's icon, the file is opened in Integra Live as expected.

Finally, Integra Live supports infinite undo and redo for all undoable actions. Undoable actions include: adding or removing tracks, blocks, modules and scenes; changing module attributes; and renaming or moving entities.

Many of these features are standard in conventional desktop applications, but some or all of them are currently missing in commonly used frameworks and programming environments for live electronic music.

6. USER TESTING

So called 'hallway testing' [13] was employed throughout the development process particularly when significant new features were added. This was made possible by basing development at a UK Conservatoire, where potential users were easy to find and willing to offer time.

Additionally, more structured lab-based testing was conducted in the later part of the project when the software had reached a semi-stable state. User testing sessions were set-up with five users following Nielsen [8]. Each session lasted 45 minutes and consisted of:

- Introductions and coffee
- Pre-questionnaire for demographic data
- 4 structured tasks focusing on specific aspects of the software
- Post-test questionnaire gathering users' evaluation and conclusions

Tests were conducted at bespoke testing facilities provided by User-lab, part of the Birmingham Institute of Art and Design in the UK. The tests were observed by both a usability researcher and an Integra developer who was available to assist with technical problems.

An evaluation of the complete findings of the user testing process is beyond the scope of this paper, however the most salient data will be presented. The post-test questionnaire showed Max/MSP to be the most commonly used software by participants seeking to achieve similar results to Integra Live. Participants were asked to rate their experience using Integra Live using a Lickert scale [6]. The results are shown in figure 7, where the red dot indicates the average score across all five participants and the yellow bar indicates the full range of answers.

Finally participants were asked to tick words from a list they felt described their experience of using the software. Words were presented in a random order for each participant in order to eliminate potential patterns emerging as a result of word ordering. The resulting word counts were then submitted to the online word-count generator Wordle[5], with all words included in the resulting word cloud and a black font with a horizontal layout. The result is shown in figure 8.

7. CONCLUSIONS

In this paper we have described the development of Integra Live, a new software application for the composition and performance of live electronic music. We have presented our requirements gathering process, and illustrated how deficiencies in existing software can be addressed by engaging users in the development process. We have described the workflow of the new interface in detail and illustrated how the interface design is intended to meet the needs of musicians and addresses a range of usability issues. Our user testing results show that Integra Live goes part way to succeeding in its goals, but still has some way to go. In general users found Integra Live to be exciting to use, and more creative and less technical than existing software for performing similar tasks. However, whilst users found Integra

[5] `http://www.wordle.net/`

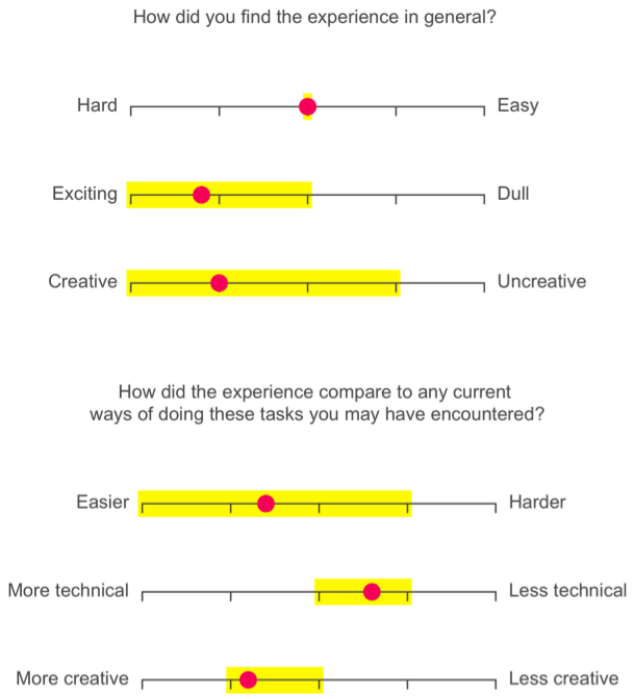

Figure 7: User testing Lickert scale results

Figure 8: Word cloud showing questionnaire word-tick response

Live 'easier' to use than existing software, they *didn't* find it 'easy'. Additionally, the software was found to be limited in terms of control-data processing functionality. Although, advanced control-data processing is possible through the scripting facility, this is clearly an unsuitable interface for entry-level use. In future work, these issues will be addressed through the addition of appropriate UI components.

8. ACKNOWLEDGMENTS

Integra Live was developed in order to meet the objectives of Integra, a 3 year EU-funded project led by Birmingham Conservatoire, following another 3-year EU-funded project, "Integra, A European Composition and Performance Environment for Sharing Live Music Technologies", both part-funded by the Culture programme of the European Commission.

9. REFERENCES

[1] K. Beck. Manifesto for agile software development. `http://agilemanifesto.org/`, Feb. 2011.

[2] J. Bullock and L. Coccioli. Towards a humane graphical user interface for live electronic music. In *Proceedings of the NIME Conference*, Pittsburgh, USA, 2009.

[3] J. Bullock and H. Frisk. The integra framework for rapid modular audio application development. In *Proceedings of the International Computer Music Conference*, Huddersfield, UK, 2011.

[4] A. Chaudhary, A. Freed, and M. Wright. An open architecture for real-time music software. In *Proceedings of the International Computer Music Conference*, Berlin, Germany, 2000.

[5] S. Krug. *DonâĂŹt Make Me Think: A Common Sense Approach to Web Usability*, chapter 1. New Rider Publishing, 2000.

[6] R. Lickert. A technique for the measurement of attitudes. *Archives of Psychology*, 50:1–55, 1991.

[7] T. Magnusson and H. M. Mendieta. The acoustic, the digital and the body: a survey on musical instruments. In *Proceedings of the NIME Conference*, New York, USA, 2007.

[8] J. Nielsen. Why you only need to test with 5 users: Alertbox. `http://www.useit.com/alertbox/20000319.html`, Feb. 2011.

[9] T. Place and T. Lossius. Jamoma: A modular standard for structuring patches in max. In *Proceedings of the International Computer Music Conference*, New Orleans, USA, 2006.

[10] T. Place, T. Lossius, and N. Peters. A flexible and dynamic c++ framework and library for digital audio signal processing. In *Proceedings of the International Computer Music Conference*, New York, USA, 2010.

[11] T. Place, T. Lossius, and N. Peters. The jamoma audio graph layer. In *Proceedings of The 13th International Conference on Digital Audio Effects*, Graz, Austria, 2010.

[12] M. Puckette. Combining event and signal processing in the max graphical programming environment. *Computer Music Journal*, 15(3):68–77, 1991.

[13] J. Spolsky. *User Interface Design for Programmers*, chapter 13. Springer, 2001.

[14] D. Zicarelli. An extensible real-time signal processing environment for max. In *Proceedings of the International Computer Music Conference*, San Francisco, USA, 1998.