MTSeq: Multi-touch-enabled CHR-based Music Generation and Manipulation

Florian Geiselhart[1], Frank Raiser[1], Jon Sneyers[2], and Thom Frühwirth[1]

[1] Faculty of Engineering and Computer Science, Ulm University, Germany
firstname.lastname@uni-ulm.de
[2] Department of Computer Science, K.U.Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Abstract. We present MTSeq, an application that combines GUI-driven multi-touch input technology with the CHR-based music generation system APOPCALEAPS and an advanced audio engine. This combination leads to an extended user experience and an intuitive, playful access to the CHR music generation system, and thus introduces CHR to musicians and other non-computer-scientists in an appropriate way. The application is fully modularized and its parts are loosely interconnected through a standard IP networking layer, so it is optionally distributable across multiple machines.

1 Introduction and Goals

In our application, we show how the Constraint Handling Rules-based (CHR [3]) music generator *APOPCALEAPS*[8] is driven via a loose-coupled, modern, and multi-touch-enabled GUI, which communicates with the CHR backend. An extended audio engine allows high-quality playback and real-time manipulation of the generated music. In combination, these components form a highly interactive music generation and manipulation environment called *MTSeq* (as an acronym of *Multi-Touch* and *Sequencer*).

The most important goal of the MTSeq application is to make the APOPCALEAPS CHR application accessible to non-computer-scientists and musicians. This creates interest in CHR in user groups who did not consider CHR as an application language by now.

Another goal is to make use of new and innovative multi-touch technology for usability improvements and a modern GUI. This technology is especially interesting if an application demands lots of parallel manipulation actions, like it is common in musical environments, e.g., for parallel effects parameter manipulation. Thus, APOPCALEAPS is a suitable candidate for use with a multi-touch interface.

Given those preconditions, a GUI design goal is to resemble the look-and-feel of common musical controllers, which are widely used among musicians. These controllers heavily depend on hardware controllers, like rotary controller, sliders, and buttons, which are good for quick and intuitive manipulation especially in live situations. They are already familiar as a control paradigm to the targeted user group on the one hand, but on the other hand, the multi-touch hardware furthermore promotes the realization of such GUI widgets.

Our paper first gives an overview of the underlying CHR music generation system *APOPCALEAPS* in Section 2. Afterwards the GUI and audio extensions that were made are described in detail in Section 3. In Section 4, we sum up the experiences gained and lessons learned throughout the implementation.

In addition to this paper, a short demonstrational video is available[3].

[3] http://www.uni-ulm.de/in/pm/forschung/themen/chr/info/downloads.html

2 APOPCALEAPS

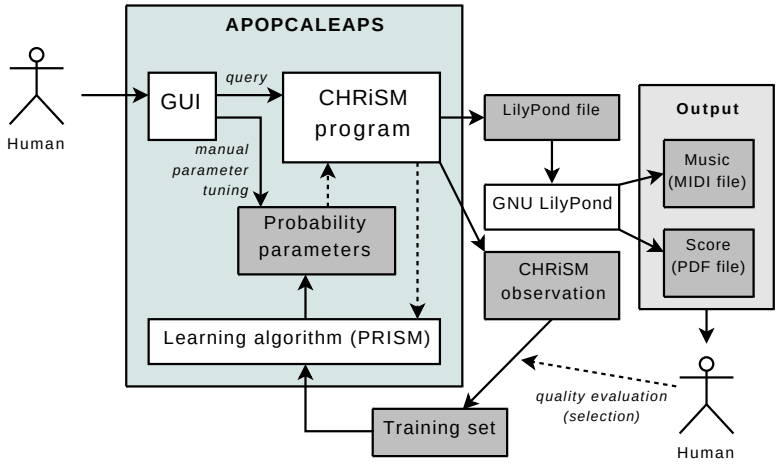

Fig. 1. An overview of the APOPCALEAPS system [8].

The core generation system of our application is formed by APOPCALEAPS[8] (an acronym for "Automatic POP Composer And LEArner of ParameterS", a music generation application which is built upon CHR[3], the probabilistic logic language PRISM[6] and its corresponding extension to CHR, called CHRiSM[9]. The remainder of this section is based on [8] and provides an overview of APOPCALEAPS.

Figure 1 gives a schematic overview of APOPCALEAPS. In the original APOPCALEAPS version, a minimal graphical user interface provides a front-end to the underlying CHRiSM program. This interface essentially allows the user to tweak an input query for the CHRiSM program, specifying some desired properties of the generated music. The default query is as follows:

```
voice(melody), shortest_duration(melody,16),
voice(bass), shortest_duration(bass,8),
voice(chords), shortest_duration(chords,8),
voice(drums), shortest_duration(drums,16),
instrument(melody,'soprano sax'),
instrument(bass,'electric bass (pick)'),
instrument(chords,'electric guitar (jazz)'),
set_range(melody,c,4,-5,16), max_jump(melody,5),
set_range(bass,c,3,-17,5), max_jump(bass,17),
chord_style(offbeat), max_repeat(melody,2),
key(major), meter(2,4), tempo(120), measures(8)
```

The above query indicates that we want a piece with four voices: melody, bass, chords and drums. The shortest possible note for the melody and drums is set to a 16th note, while for the bass and chords it is set to an 8th note. Names of MIDI instruments to be used to render the voices are given. The range of the melody is set to the interval of 5 semitones below central C to 16 semitones above central C. The biggest interval between two consecutive melody notes is set to 5 semitones. The bass has a lower range and is allowed to make bigger jumps. Chords are preferably on off-beats, and the melody should not have more than two consecutive repeated notes. The piece should be in a major key. The meter is 2/4, the tempo is 120 bpm, and the length of the piece to be generated is 8 measures.

Based on the query and probability parameters inside the CHRiSM program, CHRiSM generates output, rendered by LilyPond [5] as both a score and a MIDI file.

Additionally, the original APOPCALEAPS system in principle supports an iterative and interactive learning process, where users listen to the generated music and select the good pieces according to their own taste. The selected pieces are used as a training set for a learning algorithm that adjusts the probability parameters. The idea is that this iterative interactive process leads to a personalized music generation system.

However, due to CHRiSM not being able to deal efficiently with large output spaces yet, the learning features are computationally too expensive to be tested in practice on non-trivial examples. So for now, the CHRiSM program is driven by manual-tuned probability parameters.

The core component of the APOPCALEAPS system is, as stated above, a CHRiSM program. It consists of about 50 CHRiSM rules (about 150 lines of code). Besides the actual program, there is some auxiliary code (about 100 lines of code) and the code to write out the output in LilyPond syntax (about 150 lines of code).

The program uses 7 parametrized probabilistic experiments, which give rise to 92 probability distributions in total.

3 Multi-touch GUI and Audio Rendering

To extend the APOPCALEAPS system according to our goals, there were mainly two work areas. The first is the development of a multi-touch-enabled GUI in a suitable programming language, the second is the development of an extended audio processing system.

3.1 GUI Basics and CHR Interfacing

For the sake of rapid multi-touch UI prototyping on a limited time budget, Adobe Flash[4] was chosen as a base for GUI development. It allows a rapid and easy way of GUI widget design via vector based drawing and is suitable for multi-touch-enabled GUIs because of its programming model. Thus it is being often used in multi-touch applications (e.g.,[2,7,10] and numerous non-academic projects[5]).

The multi-touch tracking data used for control is encapsulated in Open Sound Control [1] (OSC) UDP messages following the TUIO [4] standard, and made available via a small UDP-to-TCP gateway program that Flash can connect to. We decided to re-use this existing way to communicate with the other components, as the audio processing environment is already compatible to the OSC standard, and the network-based architecture allows us to run APOPCALEAPS within a Linux virtual machine, while the GUI is running on a Windows host.

The main task to be done beneath the GUI was to develop a proxy-like program with the following features:

- Send and receive OSC messages via UDP/IP networking
- Keep a state model of all APOPCALEAPS parameters which is modifiable through OSC
- Serialize its state to a textual goal file as an input to APOPCALEAPS
- Start and control the generation and signal its end through OSC to the GUI

[4] `http://www.adobe.com/products/flash/`
[5] `http://www.nuigroup.com`

The first and second features are handled by a small Java program and existing OSC libraries. On generation, it serializes the parameter variables to a text file in the appropriate format derived from the structure of the goal handler of APOP-CALEAPS. To control the generation process, an already existing set of shell scripts from the first minimal GUI is reused. These scripts are called directly from the Java program, so it is always in control of the generation process.

This combination of components makes it possible to modify the parameters in real-time when the GUI is changed. The actual generation is ran on demand, in a synchronous way - that means the playback is stopped when the user triggers generation, and he has to wait until the process finishes. This is primarily necessary because the generation process is not done in real-time, but also because the audio processing environment needs about 5 seconds to unload the old MIDI file and to reload the new one.

3.2 GUI structure

Fig. 2. GUI Design and Sections

The basic sections of the GUI are pointed out in Figure 2. In the upper part of the GUI, section (1) represents the generator controls. The parameter values are sent to the generator proxy in real-time, but as described above, they do not come into effect until the generation is triggered by the GO button on the right.

The lower part of the GUI allows real-time manipulation of playback and effects. It consist of a tabbed group of effect controls (2), which can be wired to the X-Y-controller pad on the right (3) in many ways. The controller pad therefore exposes the number of fingers, the x and y value of fingers and the distance and angle between certain fingers as a numeric controller value. In the very bottom part of the GUI, the global transport controls can be found. They control basic playback parameters like speed, play/stop, and volume.

Interaction Design The whole interface and interaction is designed in style of a electronic musical device to allow quick access for unexperienced users. Especially the software rotary controller in our GUI follows this paradigm through a 2-finger control gesture, similar to the way a hardware rotary controller, e.g. on a mixing desk, might be used in the real world. In addition, the X-Y-controller resembles a real-world class of electronic musical devices, for example known as *KORG Kaoss Pad*[6]. However, an advantage of our solution is multi-touch support, which allows more parameters to be controlled at one time.

3.3 Advanced Audio and Effect Rendering

To improve the audio quality and to enable the use of effects, we integrated the commercial audio processing application *Bidule*[7]. Bidule provides a graph-oriented way of building audio processing chains and playing back audio data. Almost all of its parameters are remote controllable via OSC. We designed a so-called *patch*, that features a MIDI player and MIDI processing, a third-party software-based instrument for playing back the MIDI data from APOPCALEAPS in high quality, and a configurable set of effects that can be applied to the wave data before outputting it to the sound card. These measures greatly improve the overall user experience, especially when compared to the quality and possibilities of direct MIDI playback through a regular computer on-board sound card.

4 Conclusion

With the MTSeq system, we created an appealing application which helps to attract end users to APOPCALEAPS and thus, to CHR. We proposed and used a new and simplistic approach to the communication with CHR through a special proxy-like Java application, because this method allowed us to reuse wide parts of the APOPCALEAPS handling and control mechanisms with minor changes.

Using Adobe Flash, we were able to rapidly create a rich graphical UI, but with the consequence of having to communicate via a third party application (FLOSC[8]) that helps bypassing the Flash sandbox, because Flash normally is not able to create a listening network socket, which would be required for OSC communication.

Another problem we encountered resulted from the development process itself. As we only were able to access real multi-touch hardware from time to time, the main work had to be done via a TUIO simulator[9] to emulate a real multi-touch table. But the behaviour differences of real hardware compared to the simulator were significant, so major parts of the interaction concept had to be altered to maintain functionality on a real hardware table. This especially changed the way the wiring area works, and introduced a de-bouncing routine for switches and buttons.

Besides this, the application still leaves room for improvements. With an extension of APOPCALEAPS as a pseudo-real-time MIDI generation engine, the currently blocking-mode generation part of our application could work in real-time. This would enable the audio engine to play back a real-time MIDI data stream coming from APOPCALEAPS instead of generated MIDI files. The stream itself might be altered directly through the GUI parameters, which would result in a more responsive user experience and less waiting time compared to our current approach. However, this tends to lead to a complete rework of our CHR proxy application, as the basic concept doesn't support real-time parameter modification very well.

[6] `http://korg.com/product.aspx?&pd=269`
[7] `http://www.plogue.com`
[8] See `http://www.benchun.net/flosc/` and `http://code.google.com/p/flosc/`
[9] A part of the Reactivision framework (`http://reactivision.sourceforge.net/`)

This might, in turn, give rise to the (currently untested) direct Java interfacing of CHRiSM and PRISM through the external language interface of the underlying *B-Prolog* system.

Another feature that we did not implement to keep our GUI as simple as possible are controls for the probability parameters inside APOPCALEAPS. This might be worked out in a future version of MTSeq, but as the modification requires recompilation of parts of APOPCALEAPS to come into effect, this may introduce an additional waiting phase during music generation. Furthermore, a different GUI extension could be a direct and graphical manipulation of CHR rules and parameters on a score, as they represent the relations between the notes.

A last possible extension point is the extension and generalization of the OSC protocol subset between flash and APOPCALEAPS/CHR, as the current version only features the most basic parameters for communication and control of the APOPCALEAPS system. There are for example no error messages provided to the GUI if something goes wrong during the generation process.

References

1. OpenSound Control: state of the art 2003. National University of Singapore, Singapore, Singapore (2003)
2. Chen, C.H., Nien, K.H., Wu, F.G.: Design a multi-touch table and apply to interior furniture allocation. In: Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments. Lecture Notes in Computer Science, vol. 5615, pp. 13–19. Springer Berlin / Heidelberg (2009), `http://www.springerlink.com/content/4h97170251842644/`
3. Frühwirth, T.: Constraint Handling Rules. Cambridge University Press (2009)
4. Kaltenbrunner, M., Bovermann, T., Bencina, R., Costanza, E.: Tuio - a protocol for table based tangible user interfaces. In: Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005). Vannes, France (2005)
5. Nienhuys, H.W., Nieuwenhuizen, J.: LilyPond, a system for automated music engraving. In: Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003). Firenze, Italy (May 2003)
6. Sato, T.: A glimpse of symbolic-statistical modeling by prism. Journal of Intelligent Information Systems 31, 161–176 (2008)
7. Simona Vlad, R.V.C., Nicu, A.I.: Optical multi-touch system for patient monitoring and medical data analysis. In: International Conference on Advancements of Medicine and Health Care through Technology. pp. 279–282. Springer-Verlag, Berlin, Heidelberg (2009)
8. Sneyers, J., De Schreye, D.: APOPCALEAPS: Automatic music generation with CHRiSM. In: Downie, J., Veltkamp, R. (eds.) 11th International Society for Music Information Retrieval Conference (ISMIR 2010). Utrecht, The Netherlands (August 2010), submitted
9. Sneyers, J., Meert, W., Vennekens, J., Kameya, Y., Sato, T.: CHR(PRISM)-based probabilistic logic learning. In: Hermenegildo, M., Niemelä, I., Schaub, T. (eds.) 26th International Conference on Logic Programming. Edinburgh, UK (July 2010)
10. Strijkers, R., Muller, L., Cristea, M., Belleman, R., de Laat, C., Sloot, P., Meijer, R.: Interactive control over a programmable computer network using a multi-touch surface. In: Computational Science - ICCS 2009. Lecture Notes in Computer Science, vol. 5545, pp. 719–728. Springer Berlin / Heidelberg (2009), `http://www.springerlink.com/content/p2r7672p383gh124/`