ICSC 2024

7th INTERNATIONAL CSOUND CONFERENCE

September 17th – September 20th 2024

Vienna, Austria

PROCEEDINGS

Stadt Wien

DEPARTMENT OF MUSIC ACOUSTICS WIENER KLANGSTIL | IWK

mdw university of music and performing arts vienna

mdw Bild: shutterstock / riddboy

LOCATION

mdw – University of Music and Performing Arts Vienna
Anton-von-Webern-Platz 1, 1030 Vienna, Austria
Klangtheater (Sound Theater), Future Art Lab
Conference room AW K0101
Conference room AW M0107

CONFERENCE WEBSITE

https://www.mdw.ac.at/icsc2024/

ORGANISING COMMITTEE

Alex Hofmann, Sonja Stojak, Vasileios Chatziioannou, Oskar Gigele, Werner Goebl, Tim-Tarek Grund, Alessio Lampis, Titas Lasickas, Ewa Matusiak, Alexander Mayer, Montserrat Pàmies-Vilà, Paul Schuster, Dustin Zorn

PAPER REVIEW COMMITTEE

Øyvind Brandtsegg, Michael Gogins, Alex Hofmann, Tarmo Johannes, Luis Jure, Victor Lazzarini, Steven Yi, Rory Walsh

MUSIC REVIEW COMMITTEE

Michele Abondano, Jeanette C., Joachim Heintz, Oscar Pablo di Liscia, Feliz Macahis, Ghazaleh Moqanaki, Peter Plessas, Robert Rehnig, Dustin Zorn

SUPPORTED BY

Department of Music Acoustics – Wiener Klangstil (mdw – University of Music and Performing Arts Vienna)
Stadt Wien (City council of the city of Vienna)

PROGRAM EDITED BY

Alex Hofmann, Sonja Stojak
Department of Music Acoustics – Wiener Klangstil
mdw – University of Music and Performing Arts Vienna
Anton-von-Webern-Platz 1, 1030 Vienna, Austria

IMPRINT

Proceedings of the 7th International Csound Conference

edited by Sonja Stojak and Alex Hofmann

Department of Music Acoustics – Wiener Klangstil

mdw – University of Music and Performing Arts Vienna

Anton-von-Webern-Platz 1, 1030 Vienna, Austria

December 2024

DOI: 10.21939/ICSC2024

7th International Csound Conference

ICSC 2024

Table of Contents

Preface v

Conference program 1

Abstracts and program notes 7

Proceedings 23

PAPER SESSIONS 25

Sound Synthesis and Web Apps 25

Playing Csound Duets on the Web: How Compositional & Performance Goals Lead to Coding and Design Solutions
Richard Boulanger and John Ffitch 27

Frequency Modulation with Feedback in Granular Synthesis
Øyvind Brandtsegg and Victor Lazzarini 31

Creating Organic Generative Structures in Csound
Joachim Heintz 39

The Internet of Sound
Lorenzo Ballerini and Giuseppe Ernandez 45

cloud-5: A System for Composing and Publishing Cloud Music
Michael Gogins 51

GUIs and skills in Live-electronics 57

Cabbage is dead, long live Cabbage!
Rory Walsh 59

Envelope Shaper GUI for Complex Curves in Csound
Gianni Della Vittoria 63

Cordelia, crafting a method while live coding in Csound
Jacopo Greco D'Alceo 69

Csound Live Coding with Multiple Clients
Seokyeong Kim 75

Csound Expansion 81

Csound Journey in Iran
Parham Izadyar, Amin Khoshsabk and Ghazale Moqanaki 83

Using SOFA HRTF Files with Csound Binaural Opcodes
Brian Carty and Thom McDonnell 89

Bare-metal Csound
Aman Jagwani and Victor Lazzarini 95

Integrated Csound 1 101

Exploring the Expressive VR performance of Csound Instruments in Unity
Ken Kobayashi 103

Exploring Interactive Composition Techniques with CsoundUnity and Unit
Xiaomeng Zhong 109

Csound in the MetaVerse – From Cabbage to CsoundUnity and Beyond: Developing a Working Environment for SoundScapes, SoundCollages, and Collaborative SoundPlay
Strong Bear and Richard Boulanger 115

Face Tracking with CsoundUnity: Converting Smiles into Sounds
Bethanie Liu 121

Integrated Csound 2 125

Opening mind by opening architecture: analysis strategies
Francesco Vitucci, Giuseppe Silvi, Daniele Giuseppe Annese, Francesco Scagliola and Anthony Di Furia 127

Integrating Csound into Unreal Engine for Enhanced Game Audio
Albert Madrenys Planas 133

The advantages of multi-dimensional interfaces for the future of Csound
Hans Pelleboer 137

KEYNOTES 143

Frippertronics
Victor Lazzarini 145

Living Csound
Steven Yi 149

Why bother? The value(s) of an interface
Pierre-Alexandre Tremblay 151

ROUNDTABLE SESSION 153

Future developments in Csound and its community
Joachim Heintz and Alex Hofmann 155

WORKSHOP SESSION 157

Developing Csound
Steven Yi 159

INSTALLATION SESSION 161

Web Box - Trans-interactive installation for physical and web environments
Lorenzo Ballerini, Giuseppe Ernandez and Massimo Reina 163

Polyomino Interface for Pitch Lattices
Tim-Tarek Grund 169

Csound-FPGA Integration
Aman Jagwani and Victor Lazzarini 171

Csound in the MetaVerse: CsoundUnity at Berklee
Richard Boulanger 173

FERNNAH – Reading and Sound
Joachim Heintz 177

PROGRAM NOTES 179

ATT...
Joachim Heintz 181

Silence(d)
Marijana Janevska 183

Solar
Leon Speicher 185

Cstück Nr. 2 (2015)
Arsalan Abedian 187

Three words by Alejandra
Oscar Pablo Di Liscia 189

Oscillation Of Life
Jan Jacob Hofmann 191

Gendy Cloud
Serkan Sevilgen 193

Traverse: For Recorder and Electronics
Bethanie Liu 197

Caibleadh
Shane Byrne 199

REEHD
Clemens von Reusner 201

Eleven Questions (2024)
John Ffitch and *Richard Boulanger* 203

Decay
Patrick Dunne 205

Studio VII
Roberto Doati 207

Woodland Understorey
Mark Ferguson 209

"Franz Strauss – Five Etudes" (2021) for natural horn and electronics
Tarmo Johannes 211

A fashionable nightclub
Jean-Basile Sosa 213

Sievert
Jinhao Han 215

2024-ICSC (4)
Michael Gogins 217

Three Chants for Computer
Fernando Egido 221

CsoundScapes in the MetaVerse (2024)
Richard Boulanger 225

Female Child System – Imprisonment
Anthony Di Furia 229

Ordinary Rehearsals
Antonio Scarcia 231

WS Gluing Map
Juan Escudero 233

Ripples in the Fabric of Space-Time
Jon Christopher Nelson 235

List of Conference Contributors 237

Preface

© Inge Prader

The Rectorate of the mdw – University of Music and Performing Arts Vienna is glad to welcome the global Csound community to the 7th International Csound Conference at our university. The four-day conference on computer music brings together a diverse group of artists, academics, and Open-Source software developers. The goal of the conference is to foster a productive dialogue between Csound users, such as composers, performers and music students, and Csound developers, encouraging innovation and collaboration in the domains of electroacoustic music and computer music research.

The Department of Music Acoustics – Wiener Klangstil is an internationally recognized transdisciplinary research and education centre. It is renowned for its commitment to bridge the gap between scientific research and artistic practice, contributing to the advancement of both fields.

On behalf of the Rectorate, I warmly invite all attendees of the conference to the evening concerts, which will take place in the *Klangtheater* at mdw's *Future Art Lab*. These concerts promise to be one of the highlights of the conference, showcasing innovative performances and the latest advancements in computer music and live-electronics.

I'm convinced that the 7th International Csound Conference will provide you with valuable insights and inspiration for future research directions and I wish you stimulating discussions and a good time at the mdw.

Mag.a Ulrike Sych

Rector

PROGRAM:
7TH INTERNATIONAL CSOUND CONFERENCE

TUESDAY, September 17th

15:00-16:30 Session 1A: Registration + Installation Session (Ballerini, et al.)

LOCATION: Klangtheater – AW VU149

15:00 *Lorenzo Ballerini*, *Giuseppe Ernandez* and *Massimo Reina*
Web Box - Trans-interactive installation for physical and web environments

15:00-16:30 Session 1B: Registration + Installation Session (Grund)

LOCATION: Klangtheater – AW VU149

15:00 *Tim-Tarek Grund*
Polyomino Interface for Pitch Lattices

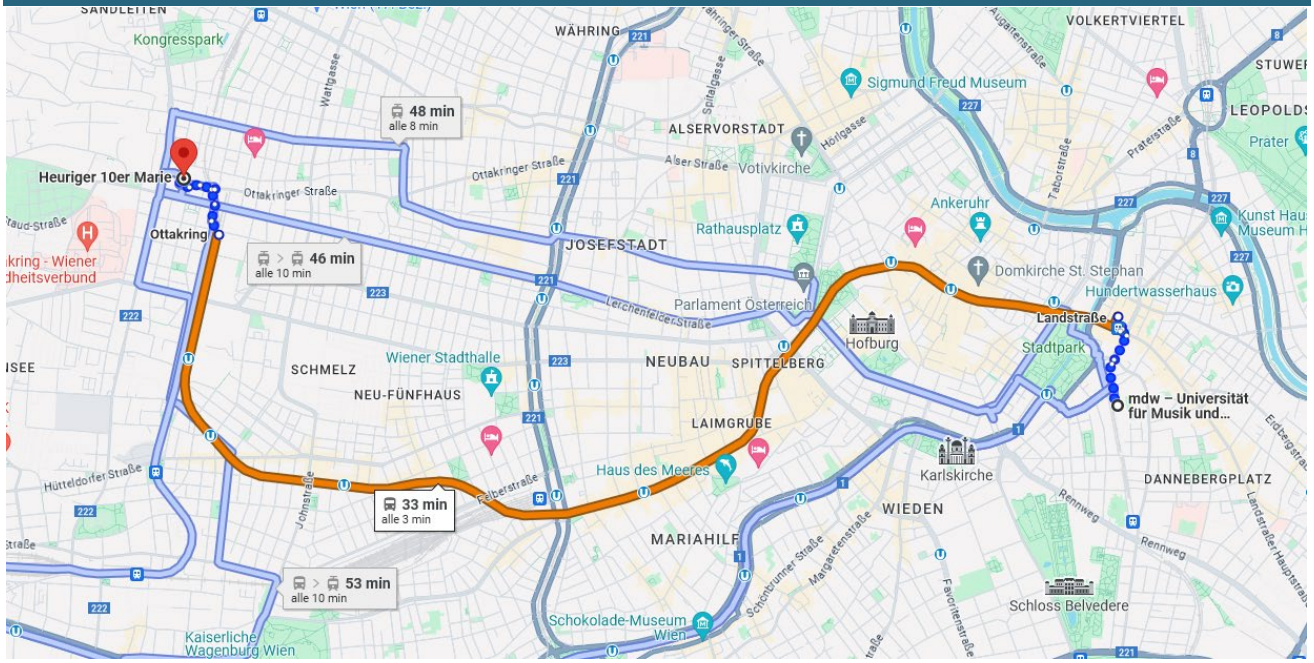
16:30-17:30 Session 2: Keynote Concert

LOCATION: Klangtheater – AW VU149
HOST: *Alex Hofmann*

16:30 Welcoming Words: Mag[a] Ulrike Sych, Rector of the mdw
Univ.-Prof. Dr.phil. Werner Goebl, Head of the Department of Music Acoustics – Wiener Klangstil

16:45 *Victor Lazzarini*
Frippertronics

19:00-22:00 Welcome dinner



Walk to the metro station Landstraße (Wien Mitte) and take the metro U3 until the metro station Ottakring. The Winery *Heuriger 10er Marie* is located in Ottakringer Straße 222-224, 1160, Vienna.
Duration: about 35 minutes.

WEDNESDAY, September 18th

08:30 Registration opens

LOCATION: AW K0101

09:00-10:40 Session 3: Sound Synthesis and Web Apps

LOCATION: AW K0101
CHAIR: *Tarmo Johannes*

09:00 *Richard Boulanger* and *John Ffitch*
Playing Csound Duets on the Web: How Compositional & Performance Goals Lead to Coding and Design Solutions

09:20 *Øyvind Brandtsegg* and *Victor Lazzarini*
Frequency Modulation with Feedback in Granular Synthesis

09:40 *Joachim Heintz*
Creating Organic Generative Structures in Csound

10:00 *Lorenzo Ballerini* and *Giuseppe Ernandez*
The Internet of Sound

10:20 *Michael Gogins*
cloud-5: A System for Composing and Publishing Cloud Music

10:40-11:00 Coffee Break – AW K0101

10:40-11:00 Session 4: Installation Session (Jagwani and Lazzarini)

LOCATION: AW K0101

10:40 *Aman Jagwani* and *Victor Lazzarini*
Csound-FPGA Integration

11:00-12:00 Session 5: Keynote Talk

LOCATION: AW K0101
CHAIR: *Alex Hofmann*

11:00 *Steven Yi*
Living Csound

12:00-13:30 Lunch Break

13:30-14:50 Session 6: GUIs and skills in Live-electronics

LOCATION: AW K0101
CHAIR: *Alex Hofmann*

13:30 *Rory Walsh*
Cabbage is dead, long live Cabbage!

13:50 *Gianni Della Vittoria*
Envelope Shaper GUI for Complex Curves in Csound

14:10 *Jacopo Greco D'Alceo*
Cordelia, crafting a method while live coding in Csound

14:30 *Seokyeong Kim*

Csound Live Coding with Multiple Clients

14:50-15:30 Coffee Break – AW K0101

15:30-17:00 Session 7: Roundtable

LOCATION: AW K0101

15:30 *Joachim Heintz* and *Alex Hofmann*

Future developments in Csound and its community

17:00-17:30 Coffee Break/Fingerfood – AW M 0107

17:00-17:30 Session 8: Installation Session (Boulanger)

LOCATION: AW M0107

17:00 *Richard Boulanger*

Csound in the MetaVerse: CsoundUnity at Berklee

17:30-19:15 Session 9: Concert + Installation Session (Grund; Ballerini, et al.)

LOCATION: Klangtheater – AW VU149

HOST: *Dustin Zorn*

- *Joachim Heintz*
  ATT... **1'49''**
- *Marijana Janevska*
  Silence(d) **6'40''**
- *Leon Speicher*
  Solar **5'00''**
- *Arsalan Abedian*
  Cstück Nr. 2 (2015) **5'03''**
- *Oscar Pablo Di Liscia*
  Three words by Alejandra **6'44''**
- *Jan Jacob Hofmann*
  Oscillation Of Life (world première) **10'44''** BREAK
- *Serkan Sevilgen*
  Gendy Cloud **8'00''**
- *Bethanie Liu*
  Traverse: For Recorder and Electronics **8'26''**
- *Shane Byrne*
  Caibleadh **7'54''**
- *Clemens von Reusner*
  REEHD **7'11''**
- *John Ffitch* and *Richard Boulanger*
  Eleven Questions (2024) **8'00''**

THURSDAY, September 19th

09:20-10:20 Session 10: Csound Expansion

LOCATION: AW K0101
CHAIR: *Joachim Heintz*

09:20 *Parham Izadyar*, *Amin Khoshsabk* and *Ghazale Moqanaki*
Csound Journey in Iran

09:40 *Brian Carty* and *Thom McDonnell*
Using SOFA HRTF Files with Csound Binaural Opcodes

10:00 *Aman Jagwani* and *Victor Lazzarini*
Bare-metal Csound

10:20-10:40 GROUP PHOTO

10:40-11:00 Coffee Break – AW K0101

10:40-11:00 Session 11: Installation Session (Heintz)

LOCATION: AW K0101

10:40 *Joachim Heintz*
FERNNAH – Reading and Sound

11:00-12:00 Session 12: Keynote Talk

LOCATION: AW K0101
CHAIR: *Tim-Tarek Grund*

11:00 *Pierre-Alexandre Tremblay*
Why bother? The value(s) of an interface

12:00-13:30 Lunch Break

13:30-17:30 Session 13: Workshop

LOCATION: AW K0101

13:30 *Steven Yi*
Developing Csound

17:30-18:00 Coffee Break/Fingerfood + Installation Session (Boulanger) – AW M0107

18:00-19:45 Session 14: Concert + Installation Session (Grund; Ballerini, et al.)

LOCATION: Klangtheater – AW VU149
HOST: *Dustin Zorn*

- *Patrick Dunne*
  Decay **1'34''**
- *Roberto Doati*
  Studio VII **7'00''**

- *Mark Ferguson*
  Woodland Understorey **4'38''**
- *Tarmo Johannes*
  "Franz Strauss – Five Etudes" (2021) for natural horn and electronics **7'30''**
- *Jean-Basile Sosa*
  A fashionable nightclub **11'00''** BREAK

- *Jinhao Han*
  Sievert **7'35''**
- *Michael Gogins*
  2024-ICSC (4) **6'10''**
- *Fernando Egido*
  Three Chants for Computer **11'15''**

FRIDAY, September 20th

09:20-10:40 Session 15: Integrated Csound 1

LOCATION: AW K0101
CHAIR: *Alex Hofmann*

09:20 *Ken Kobayashi*
Exploring the Expressive VR performance of Csound Instruments in Unity

09:40 *Xiaomeng Zhong*
Exploring Interactive Composition Techniques with CsoundUnity and Unit

10:00 *Strong Bear* and *Richard Boulanger*
Csound in the MetaVerse – From Cabbage to CsoundUnity and Beyond: Developing a Working Environment for SoundScapes, SoundCollages, and Collaborative SoundPlay
PRESENTER: *Richard Boulanger*

10:20 *Bethanie Liu*
Face Tracking with CsoundUnity: Converting Smiles into Sounds

10:40-11:00 Coffee Break + Installation Session (Heintz) – AW K0101

11:00-12:00 Session 16: Integrated Csound 2

LOCATION: AW K0101
CHAIR: *Giovanni Bedetti*

11:00 *Francesco Vitucci*, *Giuseppe Silvi*, *Daniele Giuseppe Annese*, *Francesco Scagliola* and *Anthony Di Furia*
Opening mind by opening architecture: analysis strategies

11:20 *Albert Madrenys Planas*

Integrating Csound into Unreal Engine for Enhanced Game Audio

11:40 *Hans Pelleboer*

The advantages of multi-dimensional interfaces for the future of csound

12:00-14:00 Lunch Break

14:00-15:00 Session 17: Concert

LOCATION: Klangtheater – AW VU149
HOST: *Alex Hofmann*

- *Richard Boulanger*
  CsoundScapes in the MetaVerse (2024)
- *Anthony Di Furia*
  Female Child System – Imprisonment
- *Antonio Scarcia*
  Ordinary Rehearsals
- *Juan Escudero*
  WS Gluing Map
- *Jon Christopher Nelson*
  Ripples in the Fabric of Space-Time

15:00-15:45 Closing Ceremony

LOCATION: Klangtheater – AW VU149

ABSTRACTS AND PROGRAM NOTES: 7TH INTERNATIONAL CSOUND CONFERENCE

TUESDAY, SEPTEMBER 17TH

15:00-16:30 Session 1A: Registration + Installation Session

LOCATION: Klangtheater – AW VU149

15:00

Lorenzo Ballerini, *Giuseppe Ernandez* and *Massimo Reina*

Web Box - Trans-interactive installation for physical and web environments

In our society, an illusory freedom conceals pervasive surveillance, with socioeconomic mechanisms monitoring our actions and subtly guiding our behavior. This control is exerted through advanced computer systems, especially the Web, which functions as a complex device integrating linguistic and nonlinguistic elements, regulations, and institutions to maintain capitalist power dynamics. This installation challenges the digital control system by interweaving the real and virtual worlds. At the center of the exhibition is a glowing, resonant black box, a monolithic symbol of mystery and hidden knowledge. This monolith, an archetype of the digital deity, emanates its own light and sound by absorbing and interpreting data from a dedicated web page, accessible via a QR code, allowing visitors to interact with its virtual counterpart. In turn, the monolith reacts by altering the screens of smartphones connected to the webpage, highlighting the often invisible processes of digital surveillance and social manipulation.

Through this interaction, the installation reveals how simple actions generate information streams, highlighting the pervasive and opaque nature of digital control in contemporary society.

By exploring Csound and its Web engine, we want to offer a trans-interactive experience that evokes awe and unease, prompting reflection on the influence of the digital world on our real relationships.

The monolith and its digital black box counterpart symbolize the hidden forces that shape our destinies, encouraging visitors to critically confront the pervasive surveillance of contemporary society.

15:00-16:30 Session 1B: Registration + Installation Session

LOCATION: Klangtheater – AW VU149

15:00

Tim-Tarek Grund

Polyomino Interface for Pitch Lattices

This sound installation is using Csound to explore pitch lattices. There are several online applications that allow users to explore pitch lattices. However, few tangible interfaces for this purpose exist. The polyomino interface for pitch lattices aims to bridge this gap by providing a grid of fiducial markers representing pitches that can be played by covering them with geometric shapes (polyominoes). Moving, rotating and exchanging these pieces allows users to explore pitch relations in an intuitive way.

16:30-17:30 Session 2: Keynote Concert

LOCATION: Klangtheater – AW VU149
HOST: *Alex Hofmann*

16:30

Welcoming Words:
Mag[a] Ulrike Sych, Rector of the mdw – University of Music and Performing Arts Vienna
Univ.-Prof. Dr.phil. Werner Goebl, Head of the Department of Music Acoustics – Wiener Klangstil

16:45

Victor Lazzarini

Frippertronics

This is a performance of Frippertronics, a genre of improvised electronic music started by Robert Fripp and Brian Eno in their recording No Pussyfooting of 1973. Since then, it became one a standard mode of working for Fripp and it has influenced many musicians over the years. The essence of the genre is the use of some form of a long delay line with feedback, to accummulate in layers the live performance gestures. Fripp tends to favour the use of modal melodic fragments that can be easily recognised, building interlocking patterns, but of course this is only one of the many ways to approach it. Originally, the delay lines were established using two reel-to-reel recorders, where the tape would be fed from one to the other, the distance between the machines

defining the delays. This gave a workable maximum delay of about seven seconds. Later, digital delay lines replaced these.

In my setup, I am using Csound to provide all the processing, which allows me to define as many delay lines with any useful length in any arrangement I want. The computing resources afforded by a desktop environment pose no restrictions to this, we much sooner reach the musically useful limits of the setup than the ones imposed by the system. To me, long delay lines are a complete different beast to the ordinary delays. It is possible to push the limits of stability much further into what engineers would class as not practicable, which is an interesting form of subversion of the norms.

From a musical performance perspective, we are also in a very unstable situation, which, as Fripp reflected on, is humbling for the musician. A false step and we can come crumbling down into disarray.

Siting at the keyboard with no prior idea of what is going to ensue is also very perilous. We need to be in tune with the instrument and its interface, there is no hiding. This type of performance celebrates memory, new versus old, and, foremost, the concept of entropy - all things tend to decay after a while, and so do our musical gestures.

19:00-22:00 Welcome Dinner

Winery (Heuriger)

Heuriger 10er Marie

Ottakringer Straße 222-224, 1160 Wien

See page 1.

WEDNESDAY, SEPTEMBER 18TH

08:30-09:00 Registration

LOCATION: AW K0101

09:00-10:40 Session 3:
Sound Synthesis and Web Apps

LOCATION: AW K0101
CHAIR: *Tarmo Johannes*

09:00

Richard Boulanger and *John Ffitch*

Playing Csound Duets on the Web: How Compositional & Performance Goals Lead to Coding and Design Solutions

PRESENTER: *Richard Boulanger*

Whether your musical journey begins in the family recorder quartet or in a wedding band, a college choir, or a community orchestra, making and playing music with others is one of life's greatest and most memorable pleasures. For many years now, the authors have collaborated on compositions and enjoyed performing together on concert stages in the US, Asia, and Europe. For the past six years, they have been co-writing a new set of pieces in which, over the web, they are accompanied by and interacting with a generative algorithmic computer ensemble and both playing and controlling Csound instruments on each other's laptop and in each other's home studios. The code, design, research and advice presented in this paper is the result of the realization of the most recent of these 'long-distance' Web duets — the composition "Eleven Questions". In it, the authors share how compositional goals lead to design solutions and how those design solutions steer the work in new and different directions, often leading far beyond what they had originally imagined. What is shared here are ideas, instruments and algorithms that will hopefully be of use to other Csounders wishing to travel similar creative paths.

09:20

Øyvind Brandtsegg and *Victor Lazzarini*

Frequency Modulation with Feedback in Granular Synthesis

The paper investigates audio synthesis with frequency modulation feedback in granular synthesis, comparing it with regular FM feedback. The combinations of these two classic synthesis techniques show some promising areas of exploration. As a full exploration of this potential is beyond the scope of this paper, we will rather give insight into some initial experiments and share the tools used, encouraging the reader to dive deeper into parameter combinations not yet described.

09:40

Joachim Heintz

Creating Organic Generative Structures in Csound

This paper discusses the creation of organic generative structures in Csound exemplified by a concrete artistic example. After discussing the properties of an organic generative structure the example is described in its fundamental aspects sounds, interdependency and development. Implementation details are described and shown by code examples. Finally, the open possibilities of such an artistic approach are discussed in some aspects.

10:00

Lorenzo Ballerini and *Giuseppe Ernandez*

The Internet of Sound

Integrated into our daily lives, online systems such as the Web provide essential services and support a wide range of functions and tasks. Among these, Web Audio applications have revolutionized the production, streaming, and exploration of digital audio, offering advanced tools directly accessible from web browsers without the need for third-party software installations. This paper presents the implementation of realtime convolution reverb using Csound's engine within a web page container. The source code utilizes HTML, CSS for interface styling, and JavaScript for the Csound API implementation. Through this project, our aim is to illustrate how Csound can be employed in crafting audio and multimedia devices for the web, fostering the development of versatile environments for technical and artistic exploration, as well as and for educational inclusiveness and accessibility.

10:20

Michael Gogins

cloud-5: A System for Composing and Publishing Cloud Music

The advent of the World Wide Web, adequate support for computer graphics and audio in HTML, and the introduction of WebAssembly as a low-level language and browser- hosted runtime for any number of computer language compilers, have now created an environment well suited to the online production, publication, and presentation of music, visual music, and related media at a professional standard of technical quality. A piece of music on the World Wide Web no longer need be merely a link to a downloadable soundfile or video, or even to a stream. A piece can, indeed, be its own "app" that is live code running at near native speed in the listener's Web browser.

I call this kind of music cloud music because it exists only in the "cloud," the omnipresent computing infrastructure of the Web. I argue that this creates an entirely new environment for music that, in the future, should be developed with its own social context and to function as an alternative means of disseminating music in addition to live performances, discs, streams, and downloads. Here, I present and demonstrate cloud-5, a system of Web components for producing cloud music including, among other things, fixed medium music, music that plays indefinitely, visuals that generate music, music that generates visuals, interactive music, and live coding. cloud-5 includes a WebAssembly build of the sound programming language and software synthesis system Csound, a WebAssembly build of the CsoundAC library for algorithmic composition including chords, scales, and voice-leading, the live coding system Strudel, and supporting code for menus, event handlers, GLSL shaders, and more. A cloud-5 piece thus exists as an HTML page that embeds Csound code and/or score generation code and/or Strudel code and/or GLSL code, in the context of a static Web site that can be served either locally (for composing and performing) or remotely on the World Wide Web (for publication). cloud-5 differs from related online music systems not only by incorpo- rating Csound and CsoundAC, but even more by being designed primarily as a new medium of presentation, performance, and publication.

10:40-11:00 Coffee Break – AW K0101

10:40-11:00 Session 4:
Installation Session (Jagwani and Lazzarini)

LOCATION: AW K0101

10:40

Aman Jagwani and *Victor Lazzarini*

Csound-FPGA Integration

With the development of Bare-metal Csound, embedded systems with ARM-based CPUs can now be targeted to run Csound audio programs. This installation will demonstrate the potential of this development through an interactive, generative Csound piece running on a Digilent Zybo Z7020 board, which contains a Xilinx Zynq 7000 SoC. Csound's generative and synthesis capabilities will be interfaced with motion-sensing through LIDAR sensors to capture and convert motion in any of the common spaces of the conference into varied ambient sonic results. The purpose of this installation is to create an interactive ambience for a common space and to showcase the potential and portability of Bare-metal Csound.

11:00-12:00 Session 5: Keynote Talk

LOCATION: AW K0101
CHAIR: *Alex Hofmann*

11:00

Steven Yi

Living Csound

A meditation on Csound as living software and reflections on living with this program exploring sound and music. In this talk, I will look at Csound 7, the newest generation of our software, and discuss what it offers us today as users and as a community. I will discuss where we are today, as well as short- and long-term plans, and offer some thoughts on what we can do to nurture this program to keep it vibrant and healthy for the days ahead.

12:00-13:30 Lunch Break

13:30-14:50 Session 6:
GUIs and skills in Live-electronics

LOCATION: AW K0101
CHAIR: *Alex Hofmann*

13:30

Rory Walsh

Cabbage is dead, long live Cabbage!

In April of this year, JUCE announced a new end-user license agreement. While the updated license doesn't signify the immediate demise of Cabbage in its current form, it has presented a unique opportunity to reassess the project as a whole. Consequently, a new version of Cabbage is currently under development from the ground up. The end-user experience will remain largely unchanged: the familiar Cabbage syntax will persist, users will retain access to a wide array of widgets, and they will still be able to export to all popular plugin formats. However, the bulk of the new work will occur behind the scenes. This redesigned version will feature a significantly reduced codebase. Moreover, it will leverage the power of VS Code, providing developers with more options to create modern, responsive, and dynamic user interfaces.

13:50

Gianni Della Vittoria

Envelope Shaper GUI for Complex Curves in Csound

Creating envelopes is a valuable resource for giving movement to sound. Here we present a tool that

facilitates the creation of complex envelopes thanks to a graphical interface in which the user can quickly draw the curves necessary for the most varied musical purposes. Four typical needs in the creation of the envelope are identified and discussed: the management of the general profile, the tremolo, the loop, the random component. The output product of this software will be Csound code. Designed particularly for beginners who start learning Csound, this tool makes it possible to facilitate the understanding of the envelope in the context of the parameter on which it is applied, and to provide ready-made code useful especially in conditions of very complex shapes.

14:10

Jacopo Greco D'Alceo

Cordelia, crafting a method while live coding in Csound

Cordelia emerges as a domain-specific language optimised for generating Csound and other code. Initially conceived as a live coding language, it evolved into a multifaceted tool, embodying the fluidity and the dynamic nature of contemporary composition practices. Cordelia facilitates seamless integration of diverse musical elements, from envelopes tables to tunings files. As a composer's tool, Cordelia transcends conventional boundaries, offering pathways for live coding, extension scripting within DAW environments like Reaper, translation into Csound and graphical score. Its code structure, inherited from Csound, exemplifies meticulous attention to detail, with each parameter flawlessly organised within a coherent framework.

14:30

Seokyeong Kim

Csound Live Coding with Multiple Clients

This paper introduces a Python-based TCP socket server designed for collaborative live coding sessions utilizing the Csound engine, aimed at enhancing group music creation. The server facilitates real-time, multi-client connectivity, allowing users to dynamically create and manipulate custom Csound instruments. This system is equipped with an internal loop mechanism that manages quantized events and chord transitions, providing a rhythmic backbone for musical compositions.

Participants can engage concurrently, using a suite of commands that interact intelligently with ongoing chord changes to modify specific p-fields of the csound instruments produced. This feature ensures that musical expressions are both responsive and adaptive to the evolving sonic environment. Additionally, the system offers a variety of tools that support user interactions. Users have the capability to query and identify various components such as instruments, channels, and buses within the system. This transparency facilitates an intuitive understanding of the shared musical workspace. Moreover, the architecture allows for the manipulation of loop events tied to the server's clock. Users can easily subscribe, modify, or remove their events, enabling a fluid and dynamic compositional process. By supporting direct manipulation of musical elements in a live setting, the server not only fosters individual creativity but also enhances collaborative efforts among users.

Designed as a fun and innovative project, this server is an excellent platform for both novice and experienced musicians to experiment with collaborative composition and live performance in a digital setting. It provides a playful yet robust framework for musical exploration and interaction.

14:50-15:30 Coffee Break – AW K0101

15:30-17:00 Session 7: Roundtable

LOCATION: AW K0101

15:30

Joachim Heintz and *Alex Hofmann*

Future developments in Csound and its community

Csound has undergone a significant development over the last two decades [1, 3]. This applies to the extension of the coding language and a number of different usage cases such as on embedded platforms (e.g. Raspberry Pi, BELA [4]), but also applies to the structure of open source software development in general and the inherent community effort [2]. In this roundtable we motivate a discussion between Csound developers and Csound users on the following topics, and beyond:

Csound Development
• How do the developers see the current procedure of Csound development? What is good, what is missing?
• What could be desired contributions from the users?
• What tasks need to be addressed? Who can work on these tasks?

Csound Plugins
• What is the general status on this development?
• Why are there two plugin platforms? Can they be unified?
• What is the workflow for users?
• Which jobs need to be done, and who can do these jobs?

Csound Documentation
• State of Csound Manual, FLOSS Manual, and other parts of the documentation.

17:00-17:30 Coffee Break/Fingerfood – AW M 0107

17:00-17:30 Session 8: Installation Session (Boulanger)

LOCATION: AW M0107

17:00

Richard Boulanger, Strong Bear (Hung Vo), Xiaomeng Zhong, Ken Kobayashi and *Bethanie Liu*

Csound in the MetaVerse: CsoundUnity at Berklee

This installation will showcase four projects created and programmed in CsoundUnity by Professor Richard Boulanger's Electronic Production and Design students at the Berklee College of Music in Boston. Individual players and small groups will be able to choose from and enter immersive VR, AR, and XR worlds where they can: 1. Wander through Zhong's beautiful generative Sound Garden (La forét) and play some classic Chinese instruments; 2. Design and play expressive Csound instruments in Kobayashi's Sound Lab (Laser Synth); 3. Turn a smile into a sound with Liu's Face Tracing system; or 4. Colocate to see and collaborate with multiple local and remote players as you and they create, hit, stretch, squeeze, contort, reshape, grab, pass, catch and launch Vo's "SoundOrbs" and "SoundWanders," under the stars, on the beach, over the rooftops, and under the sea (Collaging in the MetaVerse with CsoundMeta). These CsoundUnity worlds will be installed on a number of Meta Quest 2+3 MR headsets and screencast onto multiple laptops. This will allow many to explore and play simultaneously while others can watch them play as they wait for an available headset to immerse themselves in these powerful, versatile, and fun VR soundworlds.

17:30-19:15 Session 9: Concert + Installation Session (Grund; Ballerini, et al.)

LOCATION: Klangtheater – AW VU149
HOST: *Dustin Zorn*

Joachim Heintz

ATT...

A minimalist study of the motion of an acceleration / desire / grasp -> deceleration / withdrawal / leaving -> staying / steadying / lasting, and an "accompaniment" through distant, intangible chords in quirky motion. A small salute to my teacher Younghi Pagh-Paan on the occasion of her retirement from teaching in 2011.

Marijana Janevska

Silence(d)

"Silence(d)" (2020) is a piece for female voice and electronics. The idea and inspiration about this piece came from a project, where I had a task to write a 30 second piece for solo voice concerning silence and immediately a question came to my mind: How does the silence of the silenced voice sound? This silence is not relaxing, but very loud.

Leon Speicher

Solar

"Space, the final frontier.." Since my youth I was fascinated with the imaginative influence the stars have on our culture and society. All the planets of our solar system have an influence on each other and as soon as a heavy enough object enters their gravitational field, they change their behavior and pathway. Similar things

happen between us humans. We enter each others life, have an influence and then we leave (or get kicked out).

Arsalan Abedian

Cstück Nr. 2 (2015)

The text material for the recorded voice in this piece is derived from the German Wikipedia entry on the definition of border. Here an example: “Ein Beispiel für Grenzen von eindimensionalen Räumen ist die „obere“ und „untere Grenze“ in der Mathematik [...]. Umgangssprachlich wird dafür auch Grenzwert, Schwellwert oder Schranke gebraucht.” The dreamlike (or nightmarish) sound spaces of the spoken word "Grenze", which are created with the help of granular synthesis and time stretching, are presented as sound fields. In these sound spaces, forms emerge and recede, only to reappear in a different form and gestalt. This is similar to the boundaries between countries. In this context, the concept of identity is rendered meaningless. The character of the two border areas is subsumed within a spherical grey zone that simultaneously represents both the borderlands and an independent entity. The composition Cstück Nr. 2 was created using CsoundQt and features two principal sound sources: brass and voice (recorded voice: Kara Leva). It oscillates between sound and noise, creating a morphing between the sound colours and characters of voices and brass instruments. In this process, the "between", the foreign, can be seen not only as a transition, but also as a new field.

Oscar Pablo Di Liscia

Three words by Alejandra

This work is a sort of *electroacoustic poetry-landscape* based on ideas taken from three -very similar- poems by the Argentine poet Alejandra Pizarnik. In first place, there are three *portmanteaux* words (i.e., words blending the sounds and combining the meanings of others): *Errancia*, *Resolar* and *Grismente*. These three words constitute the basis of the three sections of the work, and are decomposed, time-warped and processed in several ways. In second place, the words are also combined with the sounds of three elements that were also found in the poems: wind, water and birds. The three sections become longer as the work develops, and present the material aforementioned combined in sequences more or less similar, as in a series of variations.

Jan Jacob Hofmann

Oscillation Of Life (world première)

This piece is about the generating forces of nature. To be more precise, it is about the idea of an underlying universal power that gives shape and energy to all living beings. What if there was a yet undiscovered oscillating energy beyond acoustic and electromagnetic oscillation, that gave shape, energy and interconnection to all living beings? That enabled/guided/facilitated the organisation of molecules and cells to higher organisms, beyond genetic chemical reactions and metabolism, opposed to the common increase of entropy? That creates shape like symmetry up to far more complex mathematical order, beauty out of chaos by transmitting harmonic information? What would that oscillation sound like, if we could perceive it? Would we listen? Would we be able to tune in?

Break

Serkan Sevilgen

Gendy Cloud

The "Gendy Cloud" (2022)1 is a networked, multichannel music piece that will be realized in real time by WORC, a telematic ensemble. The ensemble members could control their instruments remotely via a web interface. Any performer can control one or more instances of the software instrument based on Csound implementation of Xenakis’s GENDYN algorithm. The control parameters are limited to reduce the learning curve and increase the adaptability to the existing interfaces. However, use of stochastic processes in the instrument allows performers to create varied timbre, patterns, and textures in a multichannel diffusion system. The project was inspired by an event during "Xenakis22: The Centenary Symposium" Orestis Karamanlis utilized GENDYN (a dynamic stochastic sound synthesis algorithm conceived by Iannis Xenakis) and prepared an audio stream that conference participants can use on their mobile phones to hear in the front of the building where Iannis Xenakis was wounded. It was a touching moment that we could be able to commemorate a great composer through his work. The idea arose from the event that if it is possible to build a software instrument based on the GENDYN algorithm that leads to collaborative music-making regardless of the physical locations.

Bethanie Liu

Traverse: For Recorder and Electronics

Inspired by the composer’s own experience of battling against depression, Traverse: for Recorder and Electronics is an eight-minute electroacoustic composition depicting the journey of walking away from a place that once harbored deep shadows of sadness. In this piece, acoustic recorders and electronics echo and interact with each other to convey the intertwining memories of the past. Each step forward in the journey is met with swirling emotional disorientation, in which the state of lostness and confusion is depicted through dark atmospheric drones and brash ring modulation

sounds. All sounds are created through live improvisational melodies performed by the composer on soprano and alto recorders, then processed with a range of Csound and Cabbage plugins. Contemporary extended techniques for the recorder such as flutter tongue and sputato are also featured in the improvisational melodies. The piece eventually resolves back to the theme, depicting the composer's return to the same place after years, still agitated, but learning to be at peace with the past. The composer is the performer of the piece, and will attend the conference to perform it live if accepted.

Shane Byrne

Caibleadh

The Last Battle of Mag Tuired was fought between the Tuatha De Dannan, an ancient race of ancient Irish dieties, and their enemies, a supernatural people known as the Fomori. The leader of the Fomori, Balór na Súile Nimhe, was defeated in battle by the hero Lugh Lámhfada, resulting in the Fomorian army being cast into the depths of the sea off the coast of Ireland. Haunting voice-like calls heard in the distance across the water on still nights, known as cailbleadh, are said to be the songs of the lost Formorian spirits, exiled to beneath the waves. The idea of cailbleadh came to mind when listening to seals along the coast, their calls echoing across the cliffs and resonating in the caves, creating an almost preternatural soundscape.

Clemens von Reusner

REEHD

REEHD is not based on sounds of real instruments, but on sounds generated by physical modeling. Physical modeling allows to go beyond the limits imposed by real instruments as well as the limits imposed by human players. This can result in certain sounds no longer having any relation to known instrumental sounds. In REEHD sound objects interact as sound gestures as well as textures in a concept of composed spatial counterpoints in virtual spaces.

"But no one should be afraid that looking at signs leads us away from things; on the contrary, it leads us into the innermost of things." (Gottfried Wilhelm Leibniz, 1646-1716)

John Ffitch and *Richard Boulanger*

Eleven Questions (2024)

'Eleven Questions' (2024) is an 8-channel internet-duet with an 'ensemble' consisting of 4 'generative' computer players (the 'choir"), and 2 live ASCII players – one playing on stage in the concert hall and the other playing remotely over the WEB via OSC and ZeroTier. The remote player is projected into the concert hall via ZOOM. The live coding of the on-stage performer is projected onto another screen. Both are hearing the entire work as it is all being realized in real-time; both are sending and receiving 'text-print' messages as feedback informing each other about what motives (questions) they are selecting, what transpositions and tempi they are setting, what chords and timbres they are playing, and how they might be affecting the sounds of the computer players and each other. Over the course of the 7-minute piece, the 'tunings' of the computer harmonies and the melodic motives move from 59-tone to 12-tone. Each motivic 'question' and every note from the 'choir' comes from a discrete location and the live performers have complete control over the timing, the tempo, the register, the dynamics, and the overall mix of all the elements in the piece. As they listen to each other, and to the computer, they question and answer, accompany and lead, compliment and contradict; in some ways, "Eleven Questions" could be considered a structured internet Csound jam as it is never exactly the same, but the players are all always 'reading' from the same algorithmic 'lead-sheet'.

THURSDAY, SEPTEMBER 19TH

09:20-10:20 Session 10: Csound Expansion

CHAIR: *Joachim Heintz*

LOCATION: AW K0101

09:20

Parham Izadyar, *Amin Khoshsabk* and *Ghazale Moqanaki*

Csound Journey in Iran

Over the past decade, the growth of Csound users in Iran has had a profound impact on the music scene, not only in the realm of electronic music but also in the general music scene. This software has empowered young composers to articulate their creative visions more effectively and more easily to perform their pieces, thereby contributing to a vibrant and evolving music scene. The accessibility of Csound, its open-source nature, and the boundless creative opportunities it offers to composers have made it a favorite companion for their music. Additionally, there is a noticeable increase in composers that utilize Csound. This innovation not only benefits composers by providing them with new tools and possibilities but also introduces fresh perspectives for listeners. It is clear that many audiences are attending more and more to live electronic music performances each year, which has enriched the connection between composers and their audience. Given these observations, it is clear that Csound has had a unique and valuable influence on the contemporary Iranian music landscape. Consequently, the aim of this article is to highlight the significance of Csound in Iranian music. To better understand the widespread appeal of Csound in Iran, some questions were written for those who have experience with Csound. Their responses in following will shed light on the positive impact Csound has had on their artistic journey.

09:40

Brian Carty and *Thom McDonnell*

Using SOFA HRTF Files with Csound Binaural Opcodes

The Csound HRTF opcodes were initially written for use with a generic 'dummy head' dataset of location measurements. More recently, the field of binaural processing has enjoyed a renaissance through the proliferation of virtual loudspeaker processing. In parallel, the SOFA file format has been developed to store HRTF datasets in a defined manner. This paper discusses a method to allow the Csound HRTF opcodes to use any SOFA HRTF dataset. The outlined approach (available as a command-line tool) takes any given SOFA HRTF dataset and preprocesses it to work with the existing opcodes; it essentially stores HRTFs for each location defined in the original 'dummy head' dataset used. A rigorous interpolation algorithm is used to derive HRTFs for non-measured locations where necessary.

10:00

Aman Jagwani and *Victor Lazzarini*

Bare-metal Csound

Csound is able to target several platforms across desktop, mobile, web and embedded environments. This enables its vast audio processing capabilities to be leveraged in a wide range of sonic and musical contexts. Particularly, embedded platforms provide great portability and flexibility for users to design custom interfaces and signal processing chains for applications like installations and live performance. However, until now, embedded support for Csound was restricted to operating system-based platforms like Raspberry Pi and Bela. This paper presents our work on the development of Bare-metal Csound, extending the embedded support to ARM-based micro-controllers. We highlight the benefits and limitations of such systems and present two platforms on which we have conducted experiments - the Electrosmith Daisy and the Xilinx Zynq 7000 FPGA System-on-Chip. We also discuss potential use cases for Bare-metal Csound as well as future directions for this work.

10:20-11:00 Coffee Break – AW K0101

10:40-11:00 Session 11: Installation Session (Heintz)

LOCATION: AW K0101

10:40

Joachim Heintz

FERNNAH – Reading and Sound

The proposed event is more a performance than an installation. But with an installation it shares that visitors can come in and leave, can move closer or stay far, and take their own time. It can happen between other events in a corridor or corner, as we had it in Montevideo on the ICSC 2015. It should be noted that the text is in German; but the proposed event is not about "understanding" the text rather than experiencing the musical space.

11:00-12:00 Session 12: Keynote Talk

LOCATION: AW K0101
CHAIR: *Tim-Tarek Grund*

11:00

Pierre-Alexandre Tremblay

Why bother? The value(s) of an interface

As everyone attending this conference will know very well, creative coders have, today more than ever, a breadth of options to make music programmatically: from specialised software old and new, to toolset expanding general computer languages, many visions of what a good art-enabling coding environment cohabitate and cross-pollinate. While trends rise and fall, along the way communities wax and wane, the artworks survive as best as they could, and the artist-programmer tries to strike a balance between inspired mastery and catching up.

But is there a value to this multitude of opportunities? Are new proposals diluting energies and foci? Are there commonalities that would be better sorted once-and-for-all? What values each of these interfaces defend, consciously or not? And what about the underlying metaphors they employ to create bridges between practices and disciplines?

In this presentation, the author will muse on these questions around the design of software environments that are foundational to artistic research through creative coding. He will try to ascertain their value, the affordances and responsibilities of such enabling endeavour, through sharing his early-career personal experience of CSound, and the emergence of the FluCoMa ecosystem.

12:00-13:30 Lunch Break

13:30-17:30 Session 13: Workshop

LOCATION: AW K0101

13:30

Steven Yi

Developing Csound

This workshop introduces users to the tools, processes, and practices involved in building and developing Csound [1]. Attendees will go through a series of exercises using popular IDEs (Xcode [2], Visual Studio and editors (Visual Studio Code [4]) to build, explore, debug, and optimize the Csound codebase. The target audience is Csound users with intermediate programming experience who may be new to C/C++ development and are interested to customize Csound for their own use as well as make contributions for the benefit of the community.

Planned activities include:

- Building Csound: Understanding the build system, setting up your tools, and diagnosing issues with builds
- Tour of Csound codebase: overview of layout of codebase; walkthrough of key data structures; a guided tour of the parser, engine, opcodes, library functions, and I/O
- Debugging Csound: work through exercises using tools (unit tests, debuggers, audio editors for waveform exploration) to diagnose and fix bugs
- Optimizing Csound: working through exercises to diagnose performance issues with internal Csound code as well as Csound CSD projects using a profiler
- Beyond the Desktop: A brief discussion and walkthrough of Android, iOS, WebAssembly, and other platforms and builds
- Questions and Answers

17:30-18:00 Coffee Break/Fingerfood + Installation Session (Boulanger) – AW M0107

18:00-19:45 Session 14: Concert + Installation Session (Grund; Ballerini, et al.)

LOCATION: Klangtheater – AW VU149
HOST: *Dustin Zorn*

Patrick Dunne

Decay – An AI-assisted Electroacoustic composition

Decay is inspired by the Works of Jonty Harrison, particularly Surface Tension and EQ. The goal of the piece was to create an entire composition using a single sound source – in this case, a box of matches. The visuals were generated using Runway's text-to-video and video-to-video AI tools. The visuals were further manipulated using the Runway motion brush to distort the generated images creating abstract shapes in the process.

Roberto Doati

Studio VII

My *Studi I-VIII* are inspired by Karlheinz Stockhausen's *Klavierstücke I-VIII*. These piano works revolve around the electronic experience of Elektronische Studie I and II. If *Klavierstücke I-IV* (1952-53) represent a sort of sketches of the electronic pieces to come, Klavierstücke V-VIII (1954-55) reveal a new attention to time which at the same time 'stretch' the form according to "statistical form criteria" and allows the author to build different timbres that emerge from the constant use of resonances produced by the silent pressure of the keys.

In my studies I wanted to recreate the colour of those years' electronic sounds, especially in its main morphology, very similar to that of piano sounds, and strongly correlated to the spectrum obtained with physical models applied to audio signals produced by a set of Julia. *Studio VII* is structured as if it were a sketch of *Klavierstücke VII*. It follows its dynamics and density using three morphological typologies: fast arpeggios, long single sounds, slow arpeggios. Each sound is conceived as a *momentform*.

Mark Ferguson

Woodland Understorey

Recollections from a Cotswold woodland. Tall ash and sycamore trees in fog, heavy with condensation; leaves bending, thick drops rolling off them as a kind of half-rain. Tawny owls and pheasants, louder than expected. An evening shower moves through. It is a scene of shelter and delicate interplay, infused with the smells of damp earth.

Tarmo Johannes

"Franz Strauss – Five Etudes" (2021) for natural horn and electronics

Tarmo Johannes on "Franz Strauss – Five Etudes" (2021): "I created this piece in the summer of 2021 when Erik Alalooga, an Estonian noise artist invited me to play at a open air summer experimental music event in Tallinn. At that time, I had relatively recently started learning the natural horn as a new hobby. Considering Erik Alalooga's preference for rather harsh sounds, I wanted to combine my especially novice attempts at playing Franz Strauss's horn etudes with electronic processing, which, according to a certain algorithm, mercilessly overrides the horn's triadic passages from time to time. Additionally, there is a contrast here between the perhaps somewhat tedious regularity typical of etudes and the unpredictability of the processing."

Jean-Basile Sosa

A fashionable nightclub

A fashionable nightclub is a live electronic music performance spatialized on variable loudspeaker arrays. With this immersive creation, Jean-Basile Sosa delivers an ethereal, phantasmatic version of some of electronic musics played in American nightclubs in the 80s and 90s... It's also a reminiscence of certain spaces of social, collective and individual freedom, where marginal cultures unfold, often foreshadowing the mores and habits of tomorrow... Without ever falling into a parody of house music or techno, the project nevertheless assimilates some of the most significant characteristics of these popular musical currents: the repetition and obstinacy of the pulse, the complete abstraction of the electronic sonorities used, the regular periodicity of squares, phrases and durations, the intuitive memorization of harmonic and rhythmic cycles and loops... The project is also motivated by the ongoing development of a digital environment dedicated to musical performance and sound spatialization. Transmissible and perennial, this environment should ideally adapt to all types of audio broadcast configuration, from projected stereo to the most modern three-dimensional sound spatialization techniques such as ambisonics.

Break

Jinhao Han

Sievert

This work is inspired by nuclear decay, using Csound and sound analysis-resynthesis measures to construct an audio-visual electronic music that displays the element decay within the theoretical framework of nuclear physics. In this piece, taking the decay process of Uranium-235 to Lead-207 as an example, radioactive nuclides release a large amount of energy through a series of α and β decays, reducing their own entropy to reach a relatively stable state. In this decay process, unstable elements lower their energy by emitting high-energy particles, gradually leaving the excited state to become a stable element. This reflects my perspective of viewing the development and change of the world from a microscopic viewpoint, extending the idea that "decay" is a process in which high-energy matter gradually stabilizes through a series of destructive changes to stabilize itself, shedding uncontrollable parts, and ultimately forming a new individual with a tight and regularly stable structure.

Michael Gogins

2024-ICSC (4)

This piece is implemented using the cloud-5 system for composing, performing, and publishing electroacoustic music: fixed medium, always-on or fixed duration, visual music, interactive music, and live coding. The cloud-5 system incorporates a WebAssembly build of Csound, supporting for displaying GLSL shaders, a WebAssembly build of the CsoundAC system of algorithmic composition with facilities for automating chords and scales, and the live coding system Strudel. This particular piece uses an adaption of a ShaderToy shader that is sampled to produce scales, chords, and notes rendered with Csound, and affords interactive control over aspects of both composition and rendering.

Fernando Egido

Three Chants for Computer

This piece experiments with the concept of intrasensory synesthesia but Instead of perceiving one sensory as another we perceive a sound feature as another one So instead of hearing colors we will perceive the time as timbre or the pitch as dynamics. To do so, I use how the perception of a musical feature affects the perception of the other musical features. The perception of one parameter is determined by the other ones, especially in the threshold of perception. We can achieve this using the thresholds of perception and the way that one parameter can determine the perception of another one to make parametric interdefinitions. For example, a pulse of gains of sound that is perceived as a temporal object can be converted into a timbrical object by accelerating the velocity of the pulses. beyond 16 – 20 hertz it will be perceived as no longer as a pulse but as a pitched sound. I call this a parametric morphing in which a sound object is perceived in a way and then using changing only one feature of this sound object it is perceived around different parametric centrality.

FRIDAY, SEPTEMBER 20TH

09:20-10:40 Session 15: Integrated Csound 1

LOCATION: AW K0101

CHAIR: *Alex Hofmann*

09:20

Ken Kobayashi

Exploring the Expressive VR performance of Csound Instruments in Unity

Electronic music instruments have revolutionized musical performances. These gadgets allow musicians to perform using sound synthesis, unlocking infinite possiblities from countless algorithms, from those explored thoroughly to the cutting edge. However, as sound synthesis technologies evolve, such digital instruments must also be reimagined. An instrument that can fully utilize the capabilities of modern synthesizer technology should allow one to perform not just novel sounds, but be more expressive with their performance. This paper explores such expressiveness through an instrument created in VR, the Laser Synth.

09:40

Xiaomeng Zhong

Exploring Interactive Composition Techniques with CsoundUnity and Unity

This paper presents different techniques and systems that were used to create an interactive composition using Csound, Unity and CsoundUnity. The paper discuss the cre- ation of compositional and performative systems designed by combining the synthesis powers of Csound and the interactive game mechanisms in Unity. These systems includes: generative music with logic in C# played using Csound Instruments, trigger based control systems mimick- ing MIDI note on/off events using Unity's collision and rigidbody mechanics, transform object and controllers functioning as real-time controls like knobs and sliders. Taking advantage of both systems, it became possible to create a game-like composition la foret.

10:00

Strong Bear (Hung Vo) and *Richard Boulanger*

Csound in the MetaVerse – From Cabbage to CsoundUnity and Beyond: Developing a Working Environment for SoundScapes, SoundCollages, and Collaborative SoundPlay

PRESENTER: *Richard Boulanger*

Csound in the MetaVerse is an immersive multiplayer system built in Unity for Meta Quest XR headsets that supports new ways to interact with Csound instruments and effects. Players are collocated into shared physical or virtual spaces, either locally, playing together in the same physical space, or remotely, joining in with other players over the internet. In these VR and AR worlds, sounds appear as physical objects that players can hit, grab, stretch, squeeze or toss away while they continue sounding and wandering freely on their own. One can also 'connect' to the sounds via 'cords' and control individual or multiple parameters with buttons or physical gestures. This systems offers new ways to play with sound in time, to play with sounds in space, and to play with each other's sounds. And in this paper, we will highlight small excerpts from the code that provides the means for some of the more exciting, unique and important features that enhance the capabilities of CsoundUnity and make possible some of the uniquely powerful modes of interaction and collaboration that our Csound in the MetaVerse environment offers.

10:20

Bethanie Liu

Face Tracking with CsoundUnity: Converting Smiles into Sounds

Csound has been widely used for sound synthesis and live performance. While much exploration has been done in expanding the potential of music-making with Csound, few studies have looked into developing Csound-based music-making tools for people with physical conditions and/or disabilities. This paper presents a preliminary design and implementation of a face tracking-based musical expression system utilizing CsoundUnity's sound design capabilities for real-time

musical performance. The goal of this development is aimed towards providing alternative methods for people with limb motor impairment to express music through facial gestures. Users could control parameters of Csound instruments through facial movements such as but not limited to opening their mouths and winking. The paper will also discuss observations from user testing sessions with patients at a rehabilitation facility.

10:40-11:00 Coffee Break + Installation Session (Heintz) – AW K0101

11:00-12:00 Session 16: Integrated Csound 2

LOCATION: AW K0101
CHAIR: *Giovanni Bedetti*

11:00
Francesco Vitucci, *Giuseppe Silvi*, *Daniele Giuseppe Annese*, *Francesco Scagliola* and *Anthony Di Furia*

Opening mind by opening architecture: analysis strategies

In numerical signal processing for electroacoustic composition, the progressive loss of specific development and research environments caused by the increasing use of digital market tools has favoured the dominance of the closed-architecture audio processor model. This model, while powerful, envisions the possibility of describing output data about its perceived characteristics, but at the cost of ignoring its internal process and interacting systems, which become complex, powerful environments but closed in an inscrutable black box, a loss we must consider. Any digital signal processing technique tells a story. Just as the words of a language incorporate social, historical and technical polysemic layers, a signal processor has its own story of implementation, a gradual technological achievement with its inevitable aesthetic consequences. Through the looking-glass of literature, one can access those environments with renewed awareness by reestablishing a scientific method and an attitude to research. In this specific case, starting from the case study of Manfred Schroeder's historical reverbs, we illustrate the process of building analytical evaluation tools, as well as practical implementation, at the basis of a conscious study path.

11:20
Albert Madrenys Planas

Integrating Csound into Unreal Engine for Enhanced Game Audio

Unreal Engine is one of the most widely used game engines in the current market, thanks to its exceptional flexibility and strong graphical capabilities. Recently, the development team has introduced a new tool called MetaSounds, designed to facilitate sound synthesis, digital processing and sound design in a native way and within a node-based interface. Despite its user-friendly interface, MetaSounds still lacks certain functionalities present in older sound engines such as Csound or SuperCollider. Currently, integrating Csound into Unreal needs the use of a middleware like FMOD or Wwise, along with Cabbage to export Csound code into a VST. However, a MetaSounds node that inherently incorporates Csound, without the use of external dependencies, and with MetaSounds adaptable, intuitive, and potent graphical interface would be a significant advancement. Thanks to Unreal Engine's support for C++ implementations and enabling developers to craft their own MetaSounds nodes, it can be possible to integrate Csound within a MetaSounds node through the Csound C++ API.

11:40
Hans Pelleboer

The advantages of multi-dimensional interfaces for the future of csound

Present day micro-controllers allow many physical properties to be translated to and from the digital domain. As non-trivial sound synthesis encompasses a large number of controlling variables, the necessary properties of an effective interface are discussed. The dichotomy between the analytic approach of computer-mediated electro-acoustics and Gestalt-based integrated human perception is shown. Special emphasis is laid on the importance of simultaneous multi-modal presentation for sensory integration and the vital role played by haptic and proprioceptic feedback. Comparisons are made between the established conventions of analog electronic equipment and the relative pioneering status of computer synthesis. Three interface designs are presented, illustrating practical steps on the possible path forward and the implications these would have for csound's further development.

12:00-14:00 Lunch Break

14:00-15:00 Session 17: Concert

LOCATION: Klangtheater – AW VU149
HOST: *Alex Hofmann*

Richard Boulanger

CsoundScapes in the MetaVerse (2024)

Featuring the Unity and CsoundUnity programming and system design of Hung Vo (aka Strong Bear), *CsoundScapes in the MetaVerse* (2024) by Richard Boulanger is an 10-minute structured SoundCollage. Under the eye of the 'watcher,' whose view of the action
from within a number of AI-generated VR worlds is screencast and broadcast for the audience to see and hear, as four local and one remote 'player' wearing Quest3 XR headsets, conjure *SoundOrbs* from thin air and then strike them, stretch them, twist them, toss them, catch them, share them, steal them, clone them, replace them, and eliminate them. The SoundOrbs produce a wide range of timbres and textures and serve in a number of ways to advance the narrative of the piece. Some SoundOrbs are generative; some are explosive; some brief and momentary; some are motivic, melodic, sequential, ostinatic; some are arhythmic and others groovy. Most are synthetic, but some are sample-based. At some points in the piece, it seems like the audience is caught in the middle of a sonic food fight, whereas, at other times, they might find themselves floating in a sound cloud, or trapped in an abandoned industrial complex listening to the chaos of gasping and groaning machines; or they might find themselves gazing around a sunken underwater city listening to the singing voices of mermaids, or lost in a cave, or on the desert moon of a distant planet, or just sitting on a beach, or on a mountaintop gazing at the stars overhead listening to the music of the spheres. All of these AI-generated visual worlds compliment and reinforce the timbre, tone, temper and drama of the palettes of Csounds that each player is presented with at that point in time – when they find themselves trasported by the system to this or that location. As such, the piece is a structured improvisation in which players are presented with specific collections of SoundOrbs along their journey, each of which contains a palette (or bank) of Csounds that they can choose from, sequentially or randomly, and that they can sonically and literally reshape and transform by the movements of their hands around and through them, or by attaching 'control cables' to them and then using a variety of mapped hand gestures and button presses to more dramatically and subtly transform and modulate them. As such, the work represents a new way to compose, play, improvise, spatialize, and experience Csounds in time and shared virtual spaces.

Anthony Di Furia

Female Child System - Imprisonment

The composition attempts to tell an imaginary story through a "sound fable". A female child with beautiful eyes, she is incarcerated alone in a huge prison, completely dark and without windows. She is unable to speak, the only glimmer of communication is represented by the sound she hears by hitting one of the steel bars in her suspended room. Through this sound, transforming it into her mind, she embarks on a dreamlike journey; along the way, her imagination gains strength and, trying to limit it, builds a "sound mosaic" that slowly falls apart to gently lead her into a parallel reality, removing the emptiness of her perception, finally returning to her prison, keeping her life altered.
She doesn't fight, she just teaches who she is. And the "sound fable" continues... The composition is inspired by a recurring dream and is dedicated to my dear friend Ottavia.

Antonio Scarcia

Ordinary Rehearsals

"Ordinary Rehearsals" is an electroacoustic piece that utilizes digital techniques inspired by the traditional workflows of tape studio recording. The piece utilizes CSound for sound synthesis through sampling, articulating complex sound gestures from initially contrasting materials. These materials are designed to evolve into a dialogue, seeking moments of equilibrium. Scores are algorithmically generated within a computer algebra system, ensuring a sophisticated integration of computational precision with artistic expression. This piece intricately explores the tension and dialogue between disparate sound elements.

Juan Escudero

WS Gluing Map

From a formal point of view this work is based on a combinatorial description of the Seifert-Weber space, which is a multiconnected hyperbolic three-manifold where the faces of a dodecahedron are identified after some rotations. The construction of a random simplicial complex of the three-manifold originates from a starting triangulation or axiom. FM synthesis, plucked strings and other Csound instruments are used. The function tables are obtained from spectra of time quasicrystals and certain models of multiperiodic variable stars light curves based on analogous temporal structuring. Multiconnected manifolds are candidates for the spatial structure of the Universe. One of the consequences would be the

observation of the same part of the cosmos in different places of the sky and it appears due to the presence of a closed loop in the manifold. Some musical correspondences of this facts are explored.

Jon Christopher Nelson

Ripples in the Fabric of Space-Time

Ripples in the Fabric of Space-Time imagines a sound world filled with the "chirps" that result from two black holes colliding. As black holes collapse into one another they create a highly deformed new black hole that emits gravitational waves from its equator. These gravitational waves move up and down in frequency a few times before they die, creating "chirps." In this work aural chirps disrupt our temporal expectations, resulting in an animated soundscape filled with rapid and playful transformations between allusions to acoustic instruments, sonic environments, and percussive noises.

This composition represents the fourth movement of Nelson's six-movement acousmatic odyssey, *The Persistence of Time and Memory.*

15:00-15:45 Session 18: Closing Ceremony

LOCATION: Klangtheater – AW VU149
HOST: *Alex Hofmann*

ICSC 2024

7th INTERNATIONAL CSOUND CONFERENCE

September 17th - September 20th 2024
Vienna, Austria

PROCEEDINGS

PAPER SESSIONS

Sound Synthesis and Web Apps

Playing Csound Duets on the Web: How Compositional & Performance Goals Lead to Coding and Design Solutions

John ffitch[1] and Richard Boulanger[2]

[1]Alta Sounds
[2]Berklee College of Music
[1]jpff@codemist.co.uk
[2]rboulanger@berklee.edu

Abstract. Whether your musical journey begins in the family recorder quartet or in a wedding band, a college choir, or a community orchestra, making and playing music with others is one of life's greatest and most memorable pleasures. For many years now, the authors have collaborated on compositions and enjoyed performing together on concert stages in the US, Asia, and Europe. For the past six years, they have been co-writing a new set of pieces in which, over the web, they are accompanied by and interacting with a generative algorithmic computer ensemble and both playing and controlling Csound instruments on each other's laptop and in each other's home studios. The code, design, research and advice presented in this paper is the result of the realization of the most recent of these 'long-distance' Web duets — the composition *Eleven Questions*. In it, the authors share how compositional goals lead to design solutions and how those design solutions steer the work in new and different directions, often leading far beyond what they had originally imagined. What is shared here are ideas, instruments and algorithms that will hopefully be of use to other Csounders wishing to travel similar creative paths.

Keywords: Duet, Netsound, Improvisation, Composing, Remote Collaboration

1 The Journey Begins

Each composition is a journey. It begins with a 'simple' goal, but almost always follows a map that is missing many details and is out of date. Routes that seemed completed are actually still under construction, and almost always, the previous way is now obstructed. Even when traveling what seems to be a well-traveled path, and following a more recent map, the journey still requires invention, ingenuity, and innovation. Quickly, one discovers they must adapt and compromise, change directions, find another way, accept current limitations, and scale their expectations. Often the composition, defined by a clear and seemingly simple set of goals, leads one on a journey to a different place, a place where something unintended, something new, something beautiful, and something truly wonderful is discovered as the way brings the work into focus and the composition reveals its true self.

2 A Few Initial Goals and Solutions

Over the Internet, and using only Csound, the authors wanted to play in each other's studios, trigger note events and control each other's computers remotely, and, most importantly, we wanted to hear the changes that we were each making locally as we were making them. Our ultimate goal was to perform in concert together with one player on stage in the concert hall, and the other playing from their studio. Planning, rehearsing, and revising was done during regular video meetings held over Zoom[3] which allowed for the immediate discussion of all aspects of the system, the sounds, and the overall form of the piece as we explored, improvised, and played together.

We used Csound's OSC opcodes[2] to send events back and forth from two computers running the same Csound orchestra. We used Zoom so that we could see each other, talk to each other, share screens, record performances, share links, change things, fix things, compose and code new things, and directly collaborate. On each of our laptops, we ran the same .csd file over a ZeroTier VPN[5].

We found that with ZeroTier we could perform in any venue without being blocked by local firewalls or needing accounts on the local system.[1] Thus, we could remotely perform together in

[1] Under the "Managed Addresses" drop-down menu, ZeroTier assigns unique IP addresses for each user.

each other's studio, university classroom, or on any concert stage. Typically, one would use MIDI controllers to 'play' Csound, but we decided that via the Csound *sensekey* opcode, all the commands and controls could be assigned to ASCII keys, thus allowing the performance to be realized from just the laptop. And we determined that, with care, the controls and interaction could be coded as a stream of single characters. This allowed commands from the local and remote computers to be accepted in any order, and there were no complications from state being preserved. This limits use of integers, and restricts commands to single keys, but the simplicity was worth the restrictions.

```
gilsten OSCinit $IPORT.
;; for each key typed or received from OSC...   change the state
instr 9, control
kk init 0
kk  OSClisten  gilisten, "/11q", "i", key    ;; Listen to remote player
printk2 kk, 0, 1
if kk==1 goto new
key, kPress sense                            ;; local sense key
if changed(kPress) != 1 goto ee
;; changes
if (kPress != 1) goto ee                     ;; no new input
   printk2 key,0,1                           ;; new remote command
new:
if (key == 45) then ;; ascii '-'
   gklastspeed = gkspeed
   gkspeed *= 1.1
  ...
ee:
...
if ((key >= 49) && (key <= 57)) then         ;; ascii 49...57 '1-9'
   schedulek (key-49)+31, 0, 0, 5, gkmspeed
...
if (key == 59) then                          ;; ascii 59 ';'
   gkmspeed *= 1.1
```

Fig. 1. Integrating local and remote commands

3 Starting Points: The Initial Ideas for the Piece

Now that we had a technical solution that works; one that allowed us to 'play' each other's computer and hear the results in our own studios, we needed to make a piece. Since our previous duet [6] was all about triggering, transposing, conducting, and permutating sequences, we decided that this duet should be chordal, more ambient, and that it might be nice to have tempo control over the chord progressions, whose durations could be either constant, or each chord-tone could have a random duration so that the progressions would slide and 'morf' from one harmony to another.

And so, we started with the idea that this work was to have eight audio generators – four computer players, controlled by algorithms, playing four-note chords whose notes shifted to the next chord at various rates, and four melodic players that were controlled by the two performing humans – one in the concert hall and one remote. Given that each human had two melody emitters, we knew it would be straightforward to design them so that the players would be able to identify their contribution to the overall sound image.

After playing a bit, we decided that it might be more musically interesting, over the course of the piece, to feature tuning progressions as well as chord progressions that would go from microtonal scales to 12-tone equal-tempered scales or jump around to different tunings for different sections of the work.

All was working well, but the piece was still missing something, and so, we decided to add 9 short melodic motives with three variations each. These consisted of fixed pitches and rhythms that could be played simultaneously, and whose dynamics and tempi could be controlled globally.

After yet more rehearsing, we felt it would be nice to add a pair of motives with a fixed sequence of pitches that could be step-advanced sequentially, randomly, or in retrograde and whose rhythms and note-durations could be determined by the player. We call them *motif0* and *motifo*, and over time, their importance in the musical structure increased, especially since they had distinctive timbres and provided opportunities for interaction and dialog between the human players.

Often in rehearsals, we were so absorbed in what we were discovering and the music we were making that one found themselves very far from where the musical journey had begun. Sometimes we were lost in reverie, sometime just out there on another planet and an 8-minute journey turned into a 20-minute excursion to a dead end. And so, commands were added so that each of the players could 'return home' and bring their systems back to square one, allowing them to start again follow a different route, and try to stay on track.

4 Making it more Musical, Defining our Roles, and Making our Contributions more Obvious

The composition was finding its way, and the path was becoming clear. We have a working system now with some nice musical possibilities. But, since we are both playing from and controlling the same Csound orchestra, it was often the case that we were not sure if author1 made that sound or author2 made that sound. And so, to better clarify our individual contributions we designed a clear set of unique melodic timbres for each player. We also decided on roles.

For instance, we decided that the person in the concert hall would control more of the balance, and be more responsible for the chord and microtonal progressions, whereas the remote player would be involved more in the tempi, transpositions, and melodic motives – essentially, one would serve as the 'accompanist-conductor' and one would be the 'soloist'. Although, it was discovered in rehearsals, that when the motives were slowed down considerably, and triggered simultaneously, that they offered another layer of very beautiful 'harmonies', and that, when they were sped up significantly, they produce interesting glitch-like textures – all still contrapuntally complimented by the step-motives that could, and often did, remain at their own tempi and unique transpositions.

```
;;;;;;;;;;;;; MOTIVE SELECTION - ASCII ;;;;;;;;;;;;;;;;
if ((key >= 49) && (key <= 57)) then ;; ascii 49...57 '1-9'
  schedulek (key-49)+31, 0, 0, 5, gkmspeed
  kcnt += 1
  printf "motif %c\n", kcnt, key
         kgoto ee
endif

;;;;;;;;;;;;; MOTIVE speed ;;;;;;;;;;;;;;;;
if (key == 58) then ;; ascii 58 ':'
  gkmspeed /= 1.1
  printf "gkmspeed+ now %g\n", gkmspeed, gkmspeed
  kgoto ee
endif
```

Fig. 2. Two simple code fragments

These two solutions were helpful, but we also felt that it would be nice if, in our consoles, we could both see what commands we had just typed and, more importantly, see what commands the other player had typed, and so, a system was devised to display the selected commands on both laptops.

And finally, we needed to devise an elegant way to start and end the piece, to insert silences, to shape phrases, and control the dynamics of the melodies and the chords, and so, a variety of mutes and fades were coded into the instruments and assigned to ASCII keys.

5 Considering the Performance Space

On the laptop, in the studio, and in the classroom, the standard monitoring system is typically stereo, but in the concert hall, very often 2-channel, 4-channel, or 8-channel PA systems are available. And so, support for all three of these playback (and recording) options was also built into our system.

6 Considering the Performance Time

As one listens to the sounds they are making, and responds to the sounds and changes that their duet partner is making, it is very easy to loose one's sense of time and for a seven minute piece to turn into a seventeen minute piece, and so, a 'timer' was added to the system that reported how long one had been playing and also sent 'reminders' and 'cues' about when to complete a section or when to start a new one.

7 Recording Rehearsals

To fine-tune the system, and consider how the piece might be improved, it was important to listen carefully as we rehearsed, but even more important to have a way to listen back and review the rehearsals an evening or day later. For this, one needed to be able to record and save the master playback (the version that audiences would hear in the concert hall), and so, support for this was built into the system. Each rehearsal is saved to the hard drive with a time stamp so they cannot get confused. The opcode *fout* is used to save the sound and the time stamp is derived from *date*.

8 Conclusion: Are We There Yet?

In the end, our specific compositional ideas and goals have led to design solutions that clearly influenced the evolution, direction, and eventuality of the composition. And we did come up with answers to a number of technical and compositional questions, all of which we are happy to share with Csounders in hopes that some of these 'solutions' might be of general interest and use. They are all a part of the 2300 line Csound orchestra for *Eleven Questions*[1]. Honestly, we are never sure that a composition is ever 'done'; there is always something one could add, something one would like to change. The 'father of Csound', Barry Vercoe, told us that "nothing conjures a double-bar line like a deadline". And so, given that the deadline approaches for the premier of the piece, (and the delivery of this paper), you might notice that the title of this composition is *Eleven Questions* and not *All the Answers*. Out of respect for the audience, we humbly stop the development here, and encourage the reader to check out all the answers we do provide in the 2300 line *csd* file we have shared [1] and we will now head off into more rehearsals and more explorations as we follow some of the paths we have charted so far and continue to discover new ones along the way. Our goal now, is to make a more satisfying, musical, and meaningful experience for each other, and for the audience, as we lead them on the paths we have charted.

References

1. John ffitch and Richard Boulanger. *Eleven Questions (2024) - csd* Submitted to ICSC 2024. `https://www.dropbox.com/scl/fo/i8hhk5zx5prwhwvo074mt/AGgyLZi2WJX6C11Bu-t9XWA?rlkey=84hc0gxquwr9o3370qwqj42lu&dl=0`
2. *The Canonical Csound Reference Manual*, 6.18.0 edition, 2024. opcodes *OSClisten* and *OSCinit*.
3. Zoom `https://zoom.us/`
4. Vercoe, B.: Real-Time Csound, Software Synthesis with Sensing and Control. In: Proceedings of the International Computer Music Conference, pp. 209–211. Glasgow (1990)
5. ZeroTier `https://docs.zerotier.com/`, May 2024.
6. John ffitch and Richard Boulanger Obsession, Sep 2022. First performed at ICSC, Athlone 2022.

Frequency Modulation with Feedback in Granular Synthesis

Øyvind Brandtsegg[1] and Victor Lazzarini[2]

[1,2]Norwegian University of Science and Technology, Maynooth University
[1]oyvind.brandtsegg@ntnu.no
[2]victor.lazzarini@mu.ie

Abstract. The paper investigates audio synthesis with frequency modulation feedback in granular synthesis, comparing it with regular FM feedback. The combinations of these two classic synthesis techniques show some promising areas of exploration. As a full exploration of this potential is beyond the scope of this paper, we will rather give insight into some initial experiments and share the tools used, encouraging the reader to dive deeper into parameter combinations not yet described.

Keywords: Frequency Modulation, Granular synthesis, Particle synthesis

1 Introduction

FM synthesis is one of the classic synthesis techniques, with early explorations by James Tenney, Jean-Claude Risset and John Chowning. A theoretical description was given in [1]. FM feedback has more recently been thoroughly investigated by [2]. Frequency modulation in granular synthesis has been briefly explored by [3] but it is still a relatively lightly explored topic. Here we will look at FM with feedback within granular synthesis as it allows some new means of pitch stabilisation, poses some new problems, and enables some exciting new sonic extensions to both FM and granular synthesis domains.

2 Basis for comparison

FM feedback with oscillators and FM feedback in granular synthesis are closely related. Both techniques use a wavetable-reading oscillator to create the output waveform, and the waveform frequency is modulated by the output waveform via feedback. Granular synthesis differ from the simpler oscillator case in that the wavetable-reading process is reinitialized on every grain, and that an envelope is applied to each grain. The envelope applied to grains can be seen as a form of amplitude modulation, with the envelope shape as the modulator waveform. The reinitialization of wavetable reading on each grain also means we can have a periodic phase reset. Phase considerations can have a significant effect on feedback modulation, as we will also see later when we apply a phase delay in the modulation feedback loop.

2.1 *Basics* for comparison

To enable a comparison between the two techniques, we have attempted to create parameter settings for the granular synthesizer that as closely as possible resemble the output from a simple oscillator. In that situation, we can compare different parameter settings that apply to both synthesis models. Then, from that comparable situation, we can later apply parameter changes to the granular process that are not available in the simple oscillator model. By doing this, we can explore in specific how the granular model extends the notion of FM feedback in the granular domain. Due to length limitations, the present article will focus on mapping out the similarities.

In granular synthesis, the perceived pitch is constituted by the grain rate ([4], [5]). We have thus chosen to let the grain rate be equal to the fundamental frequency of the simple oscillator. Similarly, it makes sense in the context of comparison to set the grain frequency (reading speed of the waveform inside each grain) equal to this fundamental frequency. With the appropriate envelope, this should

allow the granular generator to generate a signal (almost) identical to the simple oscillator. The envelope needs to have a smooth fade in and out, and there need to be sufficient overlap between grains to create a constant amplitude in the output. These constraints can be fulfilled in several different ways. With the constraint that the grain rate should be equal to the fundamental frequency, the options for the envelope are more limited. We have chosen to use a grain duration of 1.5/grainrate, which means that we have a grain overlap of 66% (the first and last 1/3 of the grain will overlap with neighboring grains). We then use 1/3 of the grain duration for fade in, and equally 1/3 of the duration for fade out. In between fade in and fade out, each grain has a full power sustain period of 1/3 of the grain duration. With an equal power crossfade, we should then be able to recreate a (non granular) waveform. A sigmoid shape is used for the fade in and out of the envelope, to enable equal power crossfading between grains. Changes to the grain duration has significant impact on the resulting sound, and in this case also affects the amount of modulation. Longer grains will give higher amount of modulation, both due to the total energy inserted into the feedback signal, and the amount of time it remains active as a modulation source.

FM feedback with simple oscillators will in its simplest form induce pitch drift, as the output waveform modulate the frequency of the oscillator. This pitch drift can be partly counteracted by recent developments in FM theory ([2]), introducing an amplitude modulation component into the feedback path. We will use this as parametric variation when we explore beyond the simplest possible comparisons. The pitch drift can also partly be neutralized by adjusting the phase of the feedback modulator. In feedback modulation, adjusting the phase can be done by introducing a small delay in the feedback signal chain. In digital audio processing the smallest possible feedback delay is 1 sample, so there is inevitably a delay in the feedback in any case. Adjusting the delay time with respect to the fundamental frequency, we attain a similar effect to adjusting the phase of an oscillator which has predictable effects in regular FM synthesis. For this reason, delay times are given in fractions of a cycle of the fundamental frequency in the code examples and experiments shown below.

2.2 Examples, comparing FM feedback in the two synthesis models

In our first example we will try to set the parameters so that we get a basic similar, or comparable, sound from both synthesis models. An empirical adjustment to the modulation index for the granular method was needed to better align the two synthesis methods. The effective modulation is dependent on grain duration (as mentioned above). Moreover, the granular model seems to need a slightly different shape of evolution for the modulation index. The empirical formula for modulation index adjustment (written in Csound code here) is thus:

```
kmodindex = (kmodindex^1.8)/(kgraindur^0.7)
```

Example 1 uses an increasing modulation index over time. As can be seen in figure 1, the sidebands show a similar evolution. The first sidebands (over the first 1.5 seconds) appear ever so slightly earlier in the oscillator model, but the break to subharmonics (octaviaton) occurs slightly earlier in the granular model (at around 2.5 seconds). The transition to chaotic behavior occurs earlier in the granular model.

The synthesis models use a set of default parameters shown in listing 1.1 at the very end of the paper, parameter settings deviating from the default setting are listed directly in the figure as nondefault parameters (in between the plots for the granular and oscillator models). The plots show a spectrogram and also zoomed-in snapshots of the waveform at selected locations. The waveform display shows 4 cycles of the waveform in each snapshot, with snapshots being taken every 0.5 seconds of the sound.

In our second example, we will look at the effect of phase delay and how the granular model has a better ability to create a stable pitch regardless of FM feedback modulation artifacts. In the previous example, we used a phase delay that would minimize pitch drift. Here we will try a different phase delay. The oscillator model pitch drift sets in relatively early (before 1 second) in the example sound, while the granular model keeps a steady pitch throughout. The pitch stability of the granular model relates to the steady grain rate, as explained above. The waveform shape can be modulated without affecting the strictly periodic placement of grains.

In our third example we try to add amplitude modulation to the feedback path to minimize oscillator pitch drift, as can be learned from [2]. We keep the phase delay value used in example 2

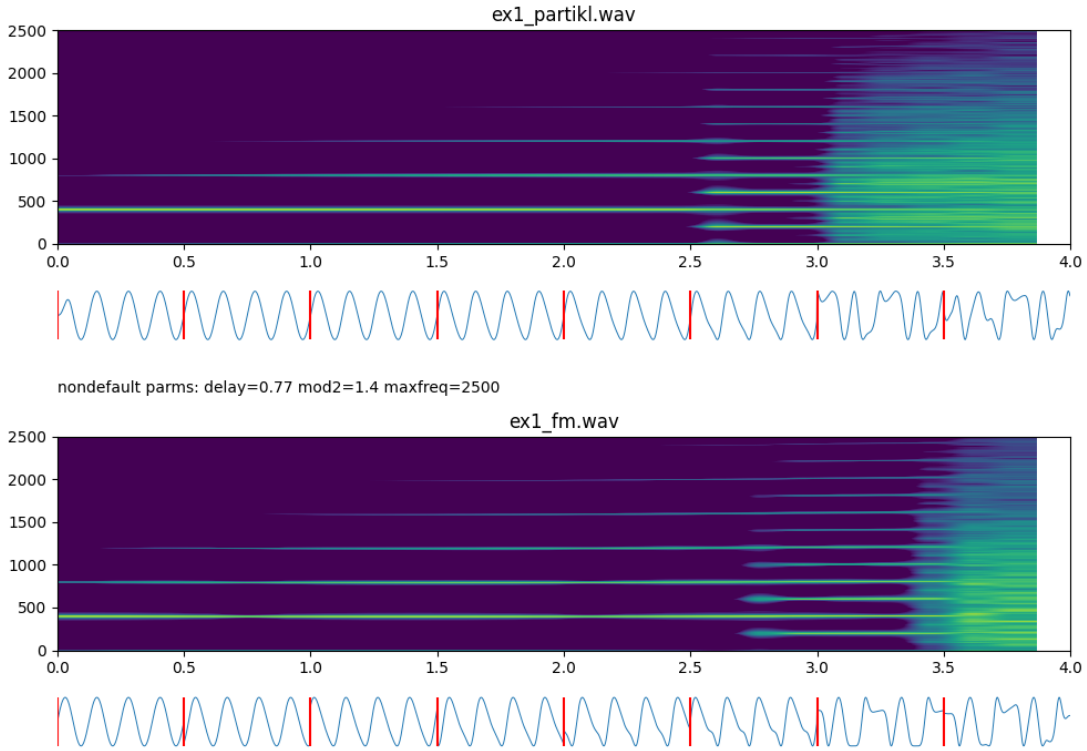


Fig. 1. Creating a similar FM feedback sound with the oscillator model and granular model

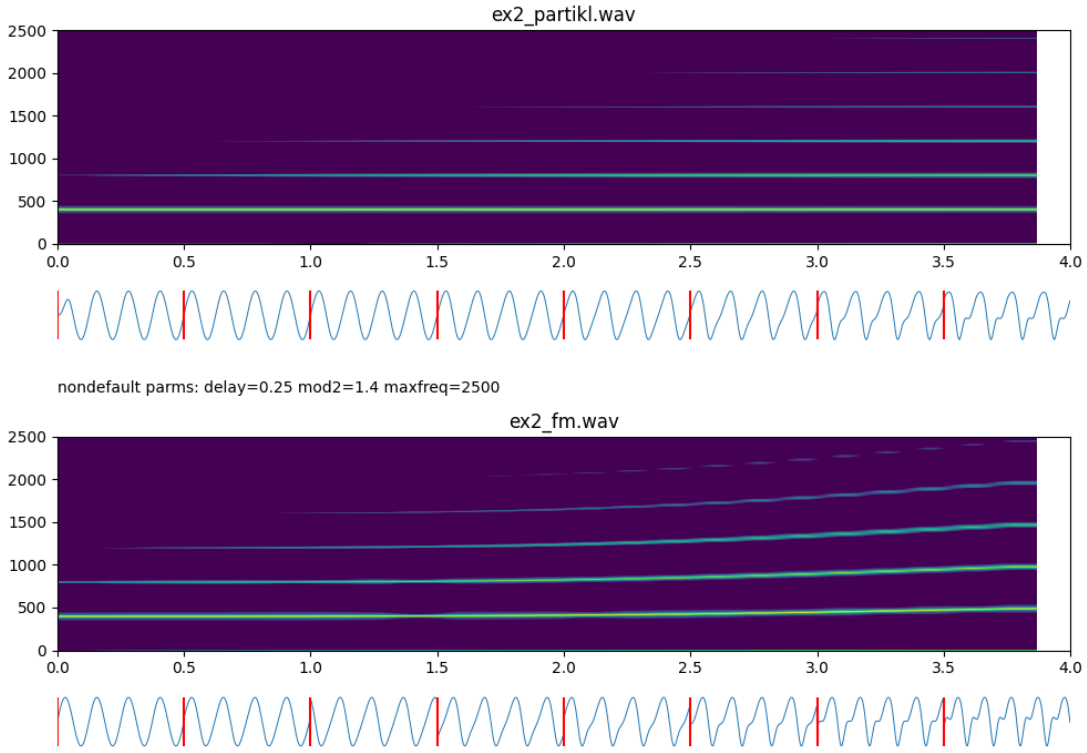


Fig. 2. Comparing pitch stability in the two synthesis models

for comparison. As seen in figure 3, the oscillator model keeps a steady pitch (up until mod index > 1.0) and also an even spacing of modulation sidebands throughout the example. The granular model displays a splitting of sidebands into subharmonics at half the fundamental frequency (at 3.3 seconds, where the mod index is approximately 1.15). From this we can assume that the AM in the feedback signal has a different effect in granular than what it has in the oscillator model. The effect of AM in the oscillator model is a more controlled situation, with pitch stability and a constant distance between modulation sidebands. In the granular model, the amplitude modulation has a lesser stabilizing effect. This might relate to the fact that the granular model already has a form of amplitude modulation inherent in the envelope for each grain.

We do note that the oscillator model is not pitch stable at modulation index above 1.0. For values beyond that range, the feedback expression cannot define the waveform uniquely as a function of time, as it has been shown in the analysis of an equivalent phase modulation arrangement [6, p.61]. The granular model is pitch stable, even if the harmonic pattern makes the fundamental pitch less prominent at high modulation indices. This could indicate that FM feedback in granular synthesis can be utilized to explore new areas of pitch stable FM feedback with high modulation indices.

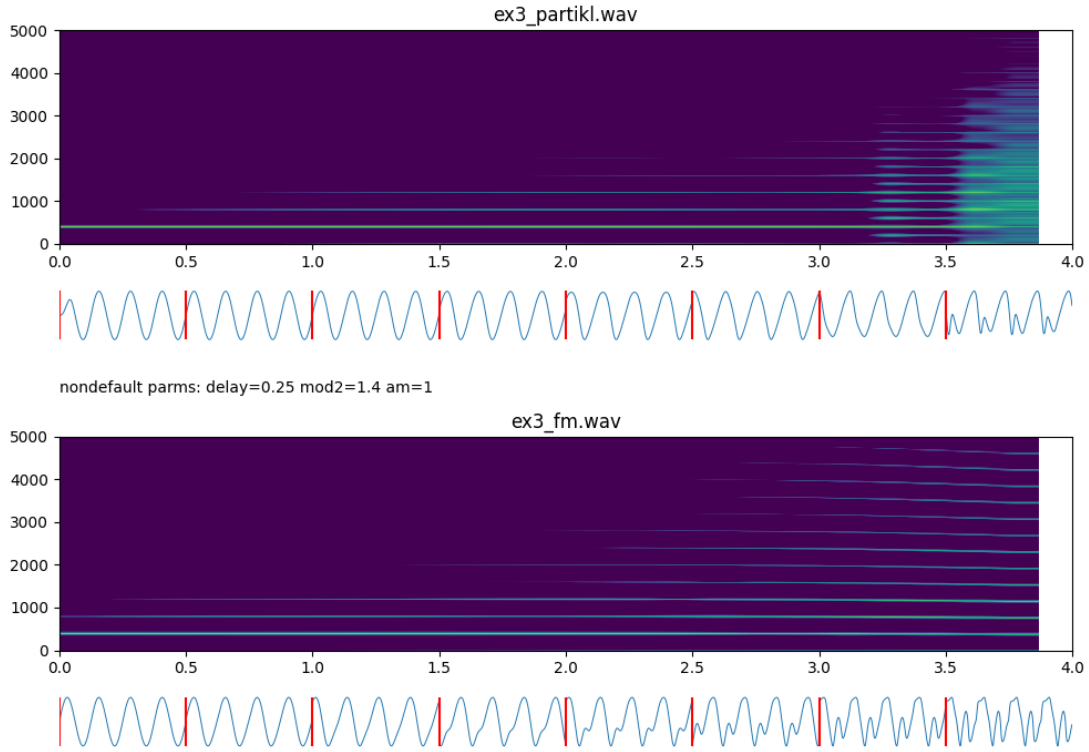


Fig. 3. Adding amplitude modulation to the feedback path

Adding a lowpass filter to the feedback path can help moderate the chaotic behavior at high modulation indices. As we see in figure 4, it also lower the amplitude of the higher partials generated. For this example we use the same delay value as in example 1, because of the better pitch stability in the oscillator model. An interesting observation is that the sidebands at half the fundamental frequency appears *earlier* with the oscillator model, as compared with the nonfiltered example (figure 1), and we also see a sudden pitch shift occuring at the same time. In the granular model, those sidebands occur at roughly the same time (same modulation index) in the unfiltered and the lowpass example provided here. Chaotic bahaviour occurs slightly later with lowpass filtering in the granular model. Pitch is stable throughout the granular example.

Another method to moderate chaotic behavior with FM feedback is to add a highpass filter in the feedback path. This will alleviate the effect of DC components resulting from the frequency

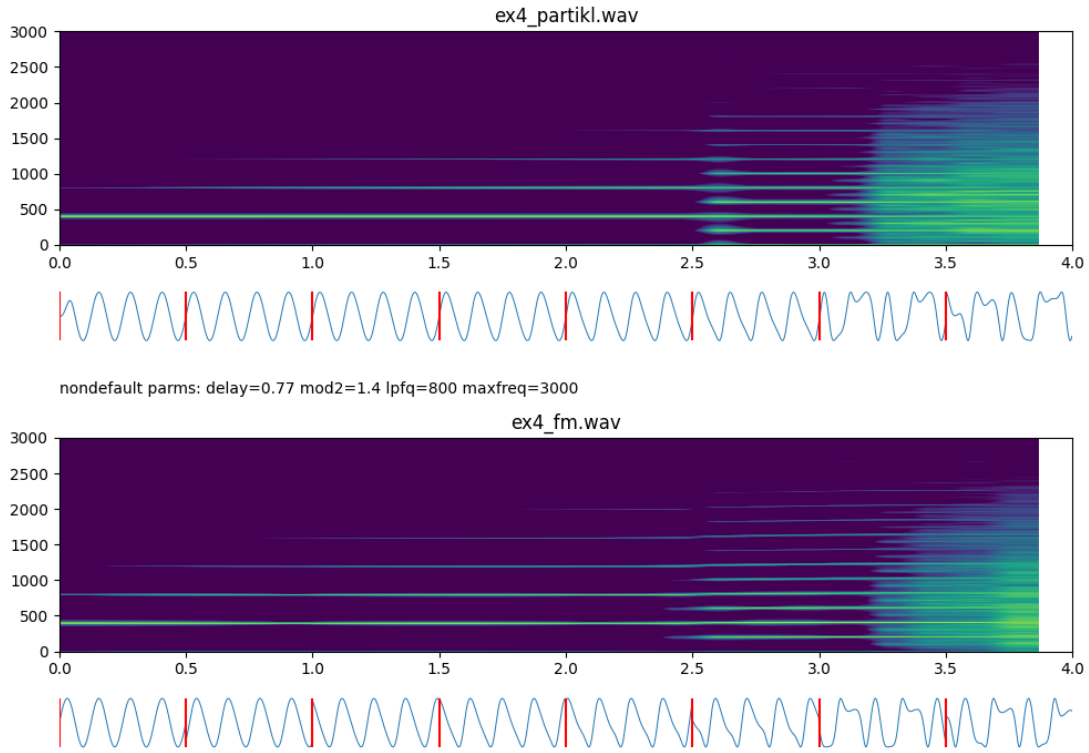


Fig. 4. Lowpass filtering the feedback path

modulation sideband at 0Hz. As can be seen in figure 5, it allows higher modulation index before chaotic behavior in the granular model. Interestingly, using the highpass filter creates clearly defined sidebands at further subdivisions (roughly 1/4) of the fundamental frequency. The upper sidebands of the oscillator model show a pitch fluctuation (not prominent in the lower sidebands).

2.3 Notes

Take note that some of the effects described are very specific to the parameter settings used. The result may be quite different with slight variation of parameter values. Also note that using a different start value for modulation index will change the behavior: Not surprisingly, the feedback behavior depends on previous values, that is, the current waveform (at any time) is a result of previous feedback, and as such, a different evolution (over time) of e.g. modulation index will lead to different sonic result (at the same modulation index value). To rephrase: the sonic result of any modulation index value depends on previous values of the modulation index. This leads to a pretty rich variety of potential outcomes, and we are currently unsure of how to relate this in a stringent manner when trying to describe specific effects of particular parameter settings. We have observed a tendency for the complexity of timbre to oscillate slightly with monotonically increasing modulation index. This phenomenon has been observed both with the granular and the oscillator model. Complexities in the sound might occur, and then subside again when increasing the modulation index further by small amounts. This in particular might be an area of further exploration, as it provides rich timbres balancing *"on the edge of chaos"* so to speak.

Regarding filters and delay, it should be mentioned that a filter in the feedback path also will induce delay. We have attempted to compensate for this filter delay in the implementation used for the examples in this paper. The filter delay is frequency dependent, but the delay compensation used here is effective for all frequencies. The required delay time has been calculated from the phase delay at the fundamental frequency of the synthesis example. One should note that some of the artifacts observed might stem from slight variations in phase delay at frequencies corresponding to sideband frequencies. Adjusting the phase delay of the feedback loop has significant effect on both the timbral

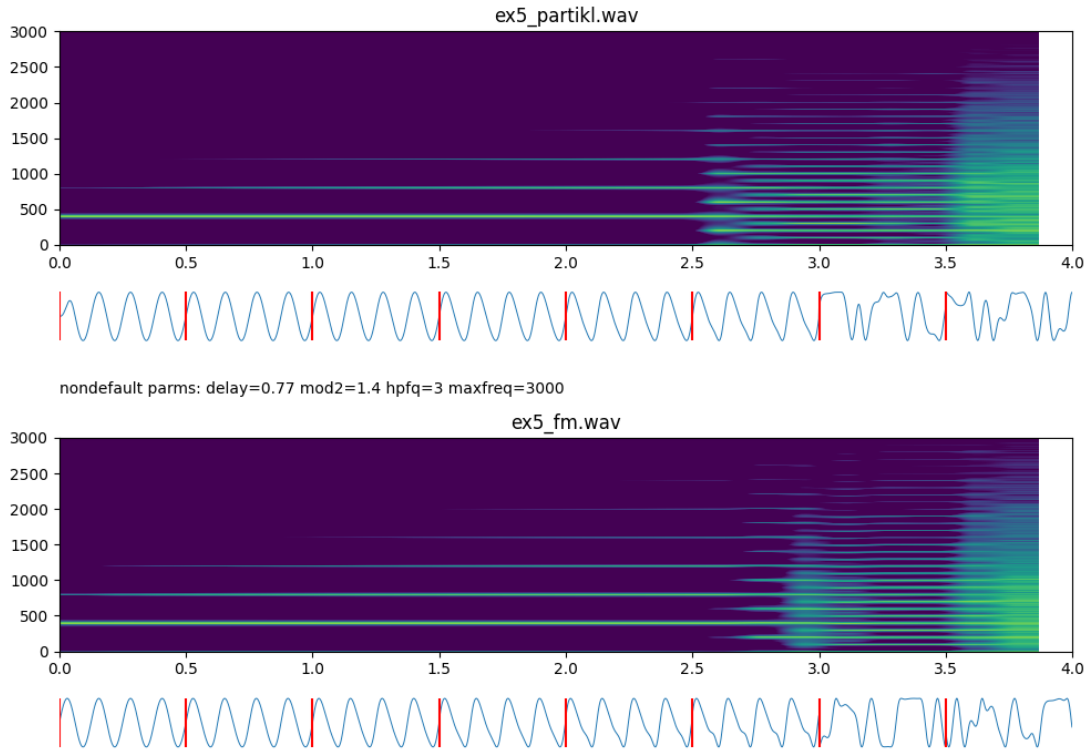


Fig. 5. High pass filtering the feedback path.

evolution and the pitch stability. Notably, it is possible to find areas of phase delay (e.g. around 0.21 and 0.77) that produce almost pitch stable results even with the traditional oscillator model. We also observe almost identical modulation behavior when adding a whole cycle to the delay time, e.g. very similar results at 1.77 as we see at 0.77. This phenomenon should be investigated further, as it seems the literature shows little previous research on the matter.

3 Running the provided code examples

The code examples for this paper can be found in a github repository at `https://github.com/Oeyvind/partikkel_fm`. There are Csound orchestra files for FM with granular synthesis and with regular oscillators. To compare the two techniques, it is useful to run both with the same set of parameters (adding only a few extra parameters for granular synthesis), and then compare the generated sound files. The repo contains python files to write score files and render sound with both techniques. This will also display spectrograms for the two generated sound files. The Python files contain default parameter settings as a starting point, and allow parameter modification via command line arguments. For example:

```
python generate_and_compare.py filename cps=200 gr.rate=200
```

will modify the fundamental frequency (cps) by setting it to 200Hz, then render sound with both synthesis techniques and display spectrograms for both generated sound files.

The github repo also contains a granular FM feedback synthesizer instrument (*partikkel_fm_feed_full.csd*) written as a Cabbage plugin with a GUI that allow the user parametric exploration of the technique.

4 Conclusion

We have explored a combination of FM feedback with granular synthesis by attempting an as close as possible comparison with regular oscillator FM with feedback. This shows some promising avenues

of further exploration, both sonically and theoretically. The tools used for exploration are available in a github repo, encouraging the reader to dive deeper into parametric combinations not yet described here.

References

1. Chowning, J. (1973). *The synthesis of complex audio spectra by means of frequency modulation.* Journal of the Audio Engineering Society, 21 (7), 527-534.
2. Lazzarini, V. and Timoney, J. (2024) *Theory and Practice of Higher-Order Frequency Modulation Synthesis.* Journal of New Music Research, 1–16. https://doi.org/10.1080/09298215.2024.2312236
3. Ervik, K. and Brandtsegg, Ø. (2013) *Combining granular synthesis with frequency modulation.* Proceedings of the 2013 Linux Audio Conference. `http://lac.linuxaudio.org/2013/papers/42.pdf`
4. Roads, C. (2001) *Mocrosound.* MIT Press. ISBN 0-262-18215-7
5. Brandtsegg, Ø. and Saue, S. and Johansen, T. (2011) *Particle synthesis–a unified model for granular synthesis.* Proceedings of the 2011 Linux Audio Conference. `http://lac.linuxaudio.org/2011/papers/39.pdf`
6. Benson, D. (1986). *Music: Mathematical Offering*, Oxford Univ. Press.
7. Walsh, R. *Cabbage - A framework for audio software development.* `https://cabbageaudio.com`
8. Lazzarini, V. et al. (2016). *Csound: A Sound and Music Computing System.* Springer.

Listing 1.1. Default parameters for the synthesis models

```
"dur" = 4 # duration of the generated sound
"amp" = -6 # overall amplitude
"cps" = 400 # fundamental frequency
"mod1" = 0 # modulation index at start of sound
"mod2" = 1.5 # modulation index at end of sound
"delay" = 0 # phase delay for the feedback modulator
"lpfq" = 21000 # lowpass filter frequency in feedback path
"hpfq" = 0 #high pass filter frequency in feedback path
"am" = 0 # enable amplitude modulation in feedback path
"gr.pitch" = 400 # grain frequency
"gr.dur" = 1.5 # grain duration relative to grain rate
"adratio" = 0.5 # attack to decay ratio of grain envelope
"sustain" = 0.33 # sustain length for the grain envelope
"index_map" = 1 # mod index empirical scaling for granular
"inv_phase2" = 0 # invert the phase of every second grain
```

Comments to the default parameters: "cps" sets the fundamental frequency of the oscillator model, and similarly sets the grain rate for the granular model. The filters in the feedback path are completely bypassed when the cutoff frequency is near the extreme setting. This switch has been set to 20kHz for the lowpass filter, and 0.1Hz for the hipass filter. The parameters "gr.pitch", "gr.dur", "adratio", "sustain", "index_map" and "inv_phase2" are only used for the granular model. The "index_map" parameter implements an empirical compensation of the effect grain duration can have on the effective modulation index. Longer grains (overlapping) will lead to a higher amplitude for the feedback signal. The "inv_phase2" parameter attempts to implement a granular equivalent of the effect that bipolar amplitude modulation can have on the feedback signal, by inverting the phase of every second grain. For this to work correctly, it also doubles the grain rate (thus also halving the grain duration), so it takes two successive grains to synthesize one duty cycle of the waveform.

Creating Organic Generative Structures in Csound

Joachim Heintz

Hochschule für Musik, Theater und Medien Hannover
joachim.heintz@hmtm-hannover.de

Abstract. This paper discusses the creation of organic generative structures in Csound exemplified by a concrete artistic example. After discussing the properties of an organic generative structure the example is described in its fundamental aspects sounds, interdependency and development. Implementation details are described and shown by code examples. Finally, the open possibilities of such an artistic approach are discussed in some aspects.

Keywords: Generative structures, Organic structures, Signals, Composition.

1 What is an Organic Generative Structure?

I call a structure *generative* when it generates sounds or events by an internal triggering algorithm. Here is a simple example for such a generative structure[1] in Csound:

```
instr Generate
  p3 = random:i(2,5)
  aEnv = transeg:a(ampdb(random:i(-30,-10)),p3,-3,0)
  outall(poscil:a(aEnv,mtof:i(random:i(72,84))))
  schedule("Generate",random:i(1,3),1)
endin
```

Each instance of this instrument calls another instance, thus generating an endless chain of sound events. It would be easy to implement tendencies and developments into it, for instance pitches becoming higher or lower, events triggered more or less often, and so on.
This structure is generative, but it is isolated from the surrounding. It does not communicate with any other structure. It depends on nothing but itself, so to say.
This can be changed by adding the ability to *sense* qualities beyond the scope of this instrument-cell. Basically this means *receiving signals* from the "world outside". The following code implements a signal named "not any more!" into the body of the instrument. If this signal is set to on (1), the instrument will stop generating events.

```
instr ReceiveSignal
  p3 = random:i(2,5)
  aEnv = transeg:a(ampdb(random:i(-30,-10)),p3,-3,0)
  outall(poscil:a(aEnv,mtof:i(random:i(72,84))))
```

[1] I use the term "structure" although in the following examples this structure is simply a csound instrument. But a structure is something more general, allowing comparisions to a cell in biology or group dynamics between persons. As explained later, the *Fernnah* structure consists of many Csound instruments and their interaction.

```
    if (chnget:i("not any more!") == 0) then
      schedule("ReceiveSignal",random:i(1,3),1)
    endif
  endin
```

This is the seed of what I call an *organic generative structure*. It communicates with the "outer world" by sending and receiving signals. Usually it consists of different units (in Csound: instruments) which have a certain job inside this organic structure.

2 The Structure for *Fernnah*

Fernnah is a literary text which I read from time to time publicly. As accompaniment for this reading I developed an organic generative structure consisting of four elements or entities, simply called A, B, C, D. Each of them eventually produce a sound of a certain type.

2.1 Sounds and Occurence

A creates violoncello-like sounds. It plays a sequence of single tones as selection of a ten tones melodical template. Each sequence is followed by a long rest.
C is kind of complement to A. It plays a sequence of chords whenever A starts its large rest.
B and *D* are both elements consisting of a group of short sounds. B sounds like paper wipes on a table; D is similar to a wine glass being hit with a metal stick. Both elements can only occur in the pauses between A or C sounds. Whether they play, or not, is determined by two factors. At first they wait for a certain time as if they were in a queue for getting a ticket for an event. Once they arrive at the ticket counter, they either get a ticket, or not. If not, they must queue again. If yes, they wait for the next pause of A or C, and then perform their event.

2.2 Interdependency

All elements have different dependencies on each other.

- A has a kind of self-movement between the sequence and the rest ("wait time" in the code). But its clock is not independent: Whenever B, C, or D are active in A's pauses, the time stops for A. So a pause of 10 seconds will become a pause of 30 seconds in case BCD are active for 20 seconds. As one extreme possibility, A would never play again if BCD continue playing all the time.
- B can only be active (after getting a ticket) if no other element is active.
- C must wait for A's long rest (wait time between the sequences).
- D can only be active if neither A nor C are playing.

Besides these dependencies concerning the occurence of the elements ABCD there are dependencies in other musical aspects, for example in pitch. A will end on another pitch than it started. This pitch is then used by C as center pitch for its chords, and by A itself as next starting pitch.

2.3 Environment and Development

All elements exist in a situative or environmental context. This environment sets certain conditions

for each of the elements. Some examples:

- For A, the selection of the melodic template is set as first and last index by the environment. The same goes for the tempo and the time frame for the long rest.
- For B, the minimum and maximum of events in each group of wipe sounds is set by the environment, together with the pitch range and the time to wait for a ticket, also in a min-max range.
- For C, the number of chords is set as minimum-maximum. As well the possible number of permutations in the chords and the amount of frayed notes in the chords.
- For D, the probability of getting a ticket after waiting is set (same for B). As well the number of notes and the pitch range, always as minimum-maximum limits for random choices.

As mentioned, the environmental parameters can change over time. As a simple example, the probability for B or D to get a ticket after queuing, can be zero at the beginning, and increase slowly. This would result in a later occurrence of B and D compared to A and C.

3 Some Implementation Details

The implementation of this organic generative structure is done in raw Csound, i.e. without any other programming language. A GUI is not necessary as all processes run by themselves, without any intervention from outside, except starting and stopping one of the three parts by pressing the space bar. These are some essential properties of the code.

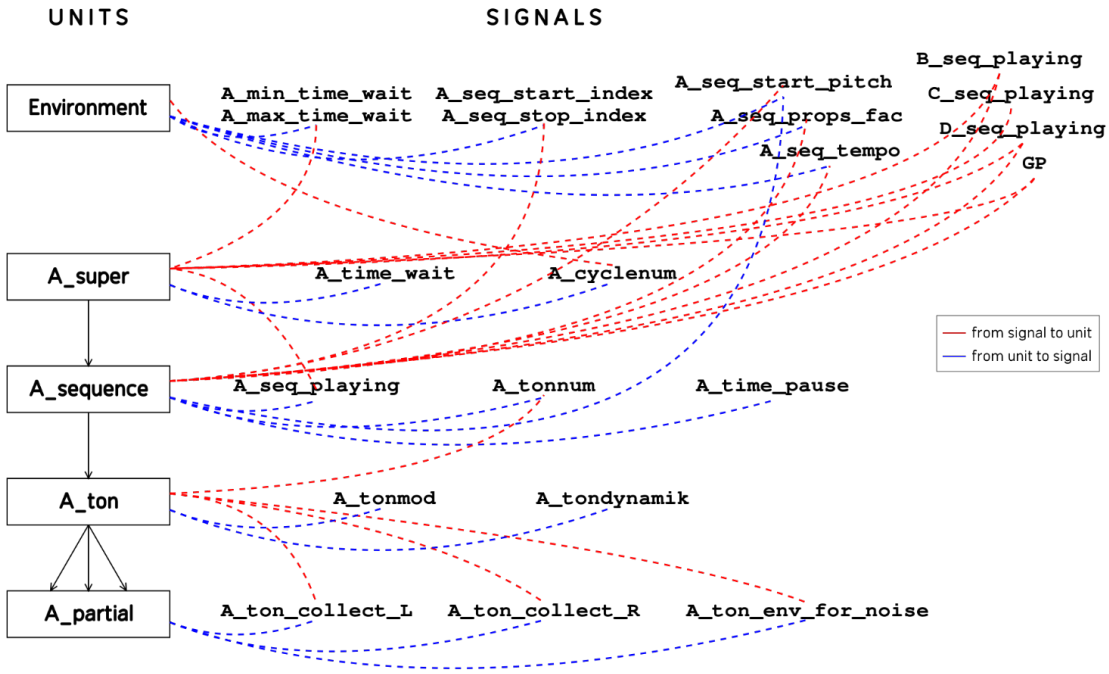
3.1 Signals

The communication inside an organic generative structure can be described as a network of signals. Signals are sent by one unit, and received and interpreted by another unit. This applies for real organic structures like bacteria or the human body, and also for a group meeting between collegues in which we send and receive signals of being bored, excited, and so on.
(As a side note: the *interpretation* of signals by the receiving units offers a very interesting field, in particular when this interpretation is wrong in the sense of functionality. In an allergy the immune system "overreacts" against small particles. In the group meeting we may "misunderstand" the smile of a member as arrogance although this person was just thinking about something nice. In art, these misinterpretations open an interesting field of suprises and unforeseeable developments.)
Sending and receiving signals in the *Fernnah* program is done via *chnset* and *chnget* in Csound. These software channels can be created inside the program itself, and are available globally. This opens infinite possibilities for signals to be read and interpreted by *any* unit; perhaps even by a unit which was not meant to do so.
This is the code for the time counter in A which only runs (= decreases the *kTime* variable by *1/kr* which is the time for one control cycle) when no other element is active:

```
if (chnget:k("B_seq_playing") == 0) &&
    (chnget:k("C_seq_playing") == 0) &&
   (chnget:k("D_seq_playing") == 0) then
  kTime -= 1/kr
endif
```

3.2 Architecture

Each of the four elements ABCD consists of different units. This is described here for A and is rather similar for BCD.
The *A_super* unit runs all the time. It watches potentially everything which happens in the program. It triggers a sequence once a new cycle starts, and counts the number of cycles.
The *A_sequence* unit is triggered by A_super. It looks for the selection of the melody template to be played, and modifies it according to some parameters, for instance to expand the intervals in the melody. It triggers the single notes for the selection, following the pauses and the tempo.
The *A_ton* unit is triggered by A_sequence and performs one of the tones to be played. Each tone consists of a number of partials and a residual (noisy) part. Different modifications are applied to always change the number of partials and their internal movement. Each tone also has a certain mode of being performed, determining envelope, vibrato, and so on.
The *A_partial* units are called by A_ton; one for each partial which is present.
Here is an overview of these units for the A element, and the signals which they receive and send.



3.3 Development and Possibilities

As mentioned earlier, the *Fernnah* reading is usually divided into three parts. Each part has a different shape and a different development. This is the *environment* in the figure above. This environment is not static but changes its parameters during the part. For instance, at the beginning of the first part B and D will never produce any sound because their probability is zero, set by the *Part_I_init* instrument:

```
chnset(0,"B_prob")
chnset(0,"D_prob")
```

Once the part has started, these two signals are increased step by step via the *Part_I_changes*

instrument. This code adds 0.2 to the previous probability for each cycle of A:

```
if (changed(chnget:k("A_cyclenum")) == 1) then
  chnset(chnget:k("B_prob")+0.2,"B_prob")
  chnset(chnget:k("D_prob")+0.2,"D_prob")
endif
```

As mentioned the possibilities to read and interpret signals are limitless. As a composer, I can just have any crazy idea about a consequence or a chain of consequences, and "just do it" by reading signals via chnget and establishing consequences via an if-else clause.

4 Result

Creating an organic generative structure in Csound is based on three parts:

1. A modular architecture between instruments which trigger each other in certain ways and under certain conditions.
2. To spread signals via chnset, and to read signals via chnget.
3. To define consequences of certain signals, or combination of certain signals, via if-then clauses.

Given these properties, coding an organic generative structure in Csound is straightforward and opens an infinite range of possible developments — developments which are based on the internal interaction of the elements in combination with environmental settings and changes.

References

Website: https://joachimheintz.net/fernnah.html
Csound Code: https://joachimheintz.de/stuecke/code/fernnah.csd
UDOs: https://joachimheintz.de/stuecke/code/fernnah_lesung.udo

The Internet Of Sound

Lorenzo Ballerini[1] and Giuseppe Ernandez[2]

[1,2]Conservatory Antonio Scontrino Of Trapany
[1]lorenzo.ballerini@constp.it
[2]giuseppeernandez@hotmail.it

Abstract. Integrated into our daily lives, online systems such as the Web provide essential services and support a wide range of functions and tasks. Among these, Web Audio applications have revolutionized the production, streaming, and exploration of digital audio, offering advanced tools directly accessible from web browsers without the need for third-party software installations.
This paper presents the implementation of realtime convolution reverb using Csound's engine within a web page container. The source code utilizes HTML, CSS for interface styling, and JavaScript for the Csound API implementation.
Through this project, our aim is to illustrate how Csound can be employed in crafting audio and multimedia devices for the web, fostering the development of versatile environments for technical and artistic exploration, as well as and for educational inclusiveness and accessibility.

Keywords: js, csound, webaudio, html, css, audio and multimedia devices, accessibility.

1 Introduction

The Web browser has become an increasingly powerful medium for developing and deploying a variety of media computing applications [1]. A notable example are Web Audio applications, which include technologies and APIs (Application Programming Interfaces) de-signed to facilitate the manipulation, sharing and play-back of audio within the Web browser; thus, also overcoming compatibility issues between different operating systems and avoiding the installation of third-party software. Furthermore, these tools can be used on PCs, tablets and smartphones.

The growing interest in these environments is clearly manifested in the openness offered by many audio software to integrate their platform-specific engines, such as Csound, Super Collider, Max MSP and Pure Data, directly within web applications.

In the following sections we will explore the possibilities offered by Csound, presenting a concise guide to developing versatile and accessible Web Audio environments. As a reference test, we decided to develop a convolution reverb, one of the processes with the highest computational cost, with the aim of analysing its performance in the network domain, also making a comparison with a similar process using the Max MSP audio engine through the RNBO development environment.

Our aim is to offer a guide to stimulate the creativity of technical and artistic work, as well as for accessible and inclusive education.

2 Js And Csound

The main elements for implementing Csound in a web container are: index.html (landing page of our application), main.csd (usual Csound source code), csound.js and csound.js.map (JavaScript library enabling the integration of Csound within web applications). All these elements must be contained in a main folder.

As index.html serves as the main entry point of the web page, all other files must be declared within it:

```
const csoundjs = "./csound.js";
const csd = "./main.csd";
```

Below is a downloadable introductory example with the elements mentioned, and its web page:

`https://csytp.github.io/webCsdTemplate`
`https://github.com/csytp/webCsdTemplate`

Here, the template and web page of the convolution reverb described in the following sections:

`https://csytp.github.io/WebCsoundVerb`
`https://github.com/csytp/WebCsoundVerb`

The code was mainly written using Vim from the command line. However, the Visual Studio Code editor, with its wide range of extensions (including web page preview and Csound syntax highlighting), can also be a good starting point for handling all the necessary processes. With these elements, it is possible to access a rich list of functionalities and abstractions within the Csound environment. Once an instance of the Csound object has been initialised, it is possible to declare file paths, load the necessary files (such as .csd files, audio files or .orc/.sco files) and start the sound engine in the web browser.

3 Writing U.I. To Simplify U.X.

Creating and linking interactive elements such as an HTML button, slider, or text box requires three steps.

3.1 HTML input object with a specific id

On the index.html the label gives a name for the object, and the input specifies his category. With the type "range" a slider is instantiated, and other properties are self-describing. The 'oninput' argument tells the web page which function is to be called from the .js file and with which arguments:

```
<label for="slider">Custom Slider</label>
<input type="range" id="slider" min="0"
max="1" value="0.5" step="0.001" oninput="
setParameter(id, value)"></input>
```

3.2 JS function that runs on a given condition

A function is needed within the .js file that connects the running instance of Csound to the web environment, passing the value received from the html page:

```
async function setParameter(channel, value)
{
if (csound) await
csound.setControlChannel(channel, value);
document.getElementById(channel +
"val").innerHTML = value;
}
```

Although the javascript code can be written inside the .html file, it is good practice to separate the code into a new .js file and call it up where necessary; this is to keep our source readable and easy to maintain. All js logic is then stored in a different file 'csoundLogic.js' and called up in the head section in the index.html file.

3.3 Csound ControlChannel in case of parameter that will be feeded on runtime

Communication is done with the Csounds opcode 'chnget' in the .csd file, which needs the variable's assigned name and type. The term "port" is used to make the scaling of values more natural:

```
kSlider = port(chnget:k("slider"),0.01,-1)
```

4 Running The Code

The application could be hosted on a repository such as GitHub to make it accessible via its web link: `https://csytp.github.io/WebCsoundVerb/`.

In cases where a reliable Internet connection is not available or the machine needs to remain offline, a local version of the application can be launched via local server; these steps are OS-specific, we provide a generic Python implementation: `python -m http.server /path/to/csoundVerb`

5 Csound For Convolution

The 'pconvolve' opcode in Csound, used within the .csd file, requires three main elements: the audio input to be processed, the impulse response file loaded into the main folder with the same name declared in the .csd file, and the partition size specifying the length of the analysis and resynthesis windows used during the convolution process. For real-time audio processing, once the user starts the engine, the browser will prompt them to select an audio interface and authorize the stream.

Other parameters can be adjusted to improve system performance. After numerous tests, we found a good compromise between overall latency, stability, and final audio quality, setting the sample rate (sr) to 48 kHz, the number of samples per control period (ksmps) to 32 samples, and the partition size (ipartitionsize) to 512 samples.

Analysis and signal processing windows introduce a delay that need to be calculated, ensuring an efficient and functional system:

```
; calculate latency of pconvolve opcode
idel = (ksmps < ipartitionsize ? ipartitionsize + ksmps : ipartitionsize)/sr
...
; delay dry signal, to align it with the convoled sig
aDryR delay (1-kmix) * aR, idel
...
outs (aDryL+aWetL) * kGain, (aDryR+aWetR)*kGain
...
; for always-on audio, an event must be executed for -1 time
schedule("Main", 0, -1 ).
```

For a detailed guide, we suggest the following web references: [2, 3, 4, 5, 6].

6 Minimal7 And Iot

John ffitch and Richard Boulanger [7] suggest that by minimising the Csound ecosystem and focusing on embedded platforms, it would be possible to create small and inexpensive devices for the Internet of Things (IoT).

These devices could make it possible to control live electronic installations from anywhere and almost in real time. Such a system could be hosted for instance on a full Linux ecosystem, which is already capable of running the main Csound build, similar to a Raspberry Pi.

Another solution could be to use two microcontrollers such as an Arduino; one microcontroller could act as a server, providing parameters via a serial protocol to the other microcontroller running a subset of Csound opcodes.

The wide range of solutions provided by these systems, integrated with the use of the Web, is a fundamental resource in a landscape increasingly oriented towards the idea of sharing and accessibility.

7 Convolution Reverb On Max MSP and Csound Web Audio

In addition to Csound many other platforms offer audio engines for web applications, including SuperCollider, Pure Data, and more recently, Max MSP with its RNBO development environment.

Developing spectral processes like convolution reverbs with impulse responses (IRs) lasting several seconds in Max MSP can be highly CPU-intensive. Achieving real-time convolution reverb with only native Max MSP objects requires partitioned convolution [8], a technique that divides the process into multiple analysis and resynthesis windows to manage computational load and latency. While Csound's pconvolve opcode already implements this efficiently, Max MSP's approach necessitates building and connecting successive analysis windows, resulting in less optimization compared to Csound.

Cipriani and Giri [9] propose a method to reduce the number of instances and improve latency and resource usage: progressively doubling the size of analysis windows using fft objects. This method enables convolution reverb implementation using only native Max MSP objects and could be compiled for Web Audio through RNBO [10].

However, this approach incurs a significantly higher computational cost. Consequently, many spectral processes in Max MSP rely on external tools, such as HISSTools [11] developed by Dr. A. Harker at the University of Huddersfield, to address computational challenges related to convolution and IRs. Using these externals precludes the use of RNBO for compiling Max patches for Web Audio.

Although comparing Max MSP and Csound directly is difficult due to differing parameters and processes, our investigations indicate that Csound provides a platform optimized for delivering high-performance native opcodes that are effective even in web environments.

8 Future Development And Conclusions

The synergy between Csound and the Web not only unlocks numerous possibilities for the development of multimedia applications but also heralds a new era in audience participation and community engagement in artistic process. We are currently embarking on the creation of a series of interactive installations aimed at redefining audience involvement as a communal and social endeavor, fostering reflections on the dynamics of relationships and encouraging active interaction with technology. Our projects, "Web Box" specifically designed for ICSC 2024, and "Transimmanency" [12], epitomize this vision of transmedia installations that unfold between real and virtual environments.

Furthermore, the duality of CS-Web is a significant opportunity in the realm of inclusive education. By providing easily accessible multimedia tools, we aspire to empower both students and educators, facilitating a more seamless and intuitive learning experience.

These initiatives represent just a glimpse of the possibilities and future progress that can be woven into the tapestry of interconnection between Csound, as well as with the similar platforms mentioned, and the Web.

References

1. Wyse, L., Subramanian, S.: The Viability of the Web Browser as a Computer Music Platform, pp. 10-23. Computer Music Journal (2013)
2. Orchestra Opcodes and Operators site, `https://csounds.com/manual/html/pconvolve.html`
3. The Absolute Vanilla Guide to Webaudio site, `https://vlazzarini.github.io/vanilla`
4. Csound on the web site, `https://kunstmusik.github.io/icsc2022-csound-web`
5. Csound Web Verb site, `https://github.com/csytp/WebCsoundVerb`
6. Webaudio Csound Samples Example site, `https://github.com/kunstmusik/webaudio-csound-samples-example`
7. Ffitch, J., Boulanger, R.: The Design and Use of Minimal7: Creating Subsets of Csound for Embedded Applications. ICSC Conference (2022)
8. Brandtsegg, Ø., Saue S., Lazzarini, V.: Live Convolution with Time-Varying Filters. Applied Sciences (2018)
9. Cipriani, A., Giri, M.: Electronic music and sound design. Theory and practice with Max 8 (Vol. 3). ConTempoNet (2021)
10. RNBO site, `https://rnbo.cycling74.com`

11. HISSTools Project site,
`https://research.hud.ac.uk/institutescentres/cerenem/projects/thehisstools`
12. Ballerini, L., Gatti, A. M.: Transimmanency: An Artistic Research Exploration of the Society of Control with Bright Resonant Objects and Web, pp. 1-5. Proceedings of the 11th International Conference on Digital and Interactive Arts (ARTECH '23). Association for Computing Machinery, New York (2023)

cloud-5:
A System for Composing and Publishing Cloud Music

Michael Gogins *

Irreducible Productions
michael.gogins@gmail.com

Abstract. This paper presents cloud-5: a toolkit for writing musical compositions, "always-on" compositions, music visualizations, animation sonifications, interactive compositions, and live-coded pieces, that play in Web browsers. A basic objective of cloud-5 is to use code running in Web browsers as *a fundamental medium of presenting music* – an alternative to physical recordings, downloads, or streams. Another objective is to provide a simplified toolkit for writing such compositions without compromising power or audio quality. The cloud-5 system includes a WebAssembly build of Csound, a WebAssembly build of the CsoundAC algorithmic composition library, and the Strudel live coding system, all integrated with a library of custom HTML elements. It is easy to install and run cloud-5. New pieces are written as Web pages without need for a build system.

Keywords: music, html5, webassembly, csound, algorithmic composition, music visualization, sonification, live coding

1 Introduction

The World Wide Web was invented for instantly sharing scientific information between scientists [1]. It was then co-opted by American businesses for the purpose of selling to consumers [2]. Along the way, it became a conduit for wholesale theft via illegal downloads of music, films, and computer games — bad enough, but the commercial and legal reactions may well have been worse [3]. Then it became the platform for social media, which provide free services and entertainment to consumers in return for selling their personal data to advertisers [4]. Indeed, most users of the Web have increasingly been funneled through Google search and various social media platforms, which are highly proprietary, far from open, and legally and politically contested [5].

And yet, at every step along this tortuous path, the inventions that created the World Wide Web, including packet-switched networking (especially TCP/IP) and the Web browser itself, loosely termed "Web standards" but in fact consisting of numerous standards from the Internet Engineering Task Force [6], the World Wide Web Consortium [7], Ecma [8], and other bodies, have remained non-proprietary, decentralized, backwards compatible, and more or less open. These are the many standards that are implemented by up to date Web browsers such as Firefox, Google Chrome, and so on [9]. In fact, driven by competitive pressures to show ever more appealing ads, the power of Web browsers has increased to the point of providing the equivalent of a game engine and an operating system, running only about 1.5 to 2.5 times as slowly as native C code [10].

I believe the establishment of Web standards will be seen in the future as one of the most fortunate events of our age, because they preserve essential freedoms in the face of a remarkable (and at times illegal) level of skilled greed. (I remain wary, however, that private interests may end up hijacking these standards and making them less open.)

The advent of the World Wide Web, adequate support for audio and computer graphics in Web pages, and the introduction of WebAssembly as a browser-hosted runtime for many computer language compilers [11], have created an environment suited to the *online* production and presentation of music, animated graphics, and related media at a *professional* standard of technical quality. For example, audio resolution in cloud-5 is the same as WebAudio: 48,000 frames per second of float samples in 128 frame blocks [12]. Although some studio software may offer even higher resolution, cloud-5 is within the recognized range of professional audio production quality [13] [14].

* I thank Dan Derks for introducing me to Tidal Cycles and Felix Roos for answering Strudel questions.

Hence, a piece of music on the World Wide Web no longer need be merely a link to a downloadable soundfile or video, or even to a stream. A piece can, indeed, be its own "app" that is live code running at near native speed in the listener's Web browser. I call this kind of music *cloud music* because it exists only in the "cloud," the omnipresent computing infrastructure of the Web. I argue that this has created an entirely new environment for music that, in the future, should be developed with its own social context and to function as an alternative means of disseminating music in addition to live performances, physical recordings, downloads, and streams.

For examples of cloud music, one may look to Generative.fm [15], WebSynth [16], Gibber [17], the Strudel live coding system [18], or my own compositions produced using the subject of this paper, cloud-5 [19]. Other such systems also integrate generative artificial intelligence, but in my view that is a separate topic deserving separate treatment.

Here, I present and demonstrate *cloud-5*, a system of Web components for producing cloud music including, among other things, fixed medium music, music that plays indefinitely, visuals that generate music, music that generates visuals, interactive music, and live coding. cloud-5 includes a WebAssembly build [20] of the sound programming language and software synthesis system Csound [21] [22] [23] [24], a WebAssembly build [20] of the CsoundAC library for algorithmic composition including chords, scales, and voice-leading [25] [26], the live coding system Strudel [18], and supporting code for menus, event handlers, GLSL shaders, and more. A cloud-5 piece thus exists as an HTML page that embeds Csound code and/or score generation code and/or Strudel code and/or GLSL code, in the context of a static Web site that can be served either locally (for composing and performing) or remotely on the World Wide Web (for publication).

cloud-5 differs from other online music systems: It is not an online development system. It is not a social medium, or a standalone Web site. It is designed primarily as *an online medium of presentation* that prioritizes the audience experience and does not compromise musical capabilities, runtime performance, or ease of use.

2 Uses

Composition The cloud-5 system includes all the astonishing capabilities already built into every standard Web browser; a WebAssembly build of Csound; a WebAssembly build of the CsoundAC algorithmic composition library; the Strudel live coding system which can render audio using either Csound or its own built-in sampler and synthesizer; and any other module that will run in a Web browser. All this is completely cross-platform. That makes cloud-5 *very* easy to install and configure on *any* supported platform: unzip it somewhere and run a local Web server there.

Fixed Pieces These are similar to fixed medium pieces of electroacoustic music. When the user starts the piece, it plays until the score ends.

Always-On Pieces Similar to fixed pieces; but once started, always-on pieces play indefinitely, and may use randomization or chaos to avoid repetition.

Interactive Pieces Similar to fixed pieces or always-on pieces; but once the piece is started, the user interface has controls with which the user may steer aspects of the composition or rendering.

Live Coding Similar to interactive pieces; but the user has *complete control* over the composition in the Strudel REPL, and can create entirely new compositions or engage in lengthy improvisations.

Music Visualization Similar to the other pieces above, but a GLSL shader displays a visualization controlled by the audio on the background of the piece; this visualization can be made full screen.

Sonification of Animations Similar to music visualization, but the video buffer is periodically downsampled or otherwise processed to produce Csound events that are rendered in real time.

Network Pieces Any of the above can fetch resources from the Internet, or be controlled remotely.

3 Design

The cloud-5 user interface (Figure 1) consists of a menu running across the top of the page. Clicking a button can start or stop performance, or show/hide various overlays that fill the page.

The system is constructed as a library of HTML custom elements, which encapsulate code and even some styling within each custom element. That makes it easier for users not familiar with the

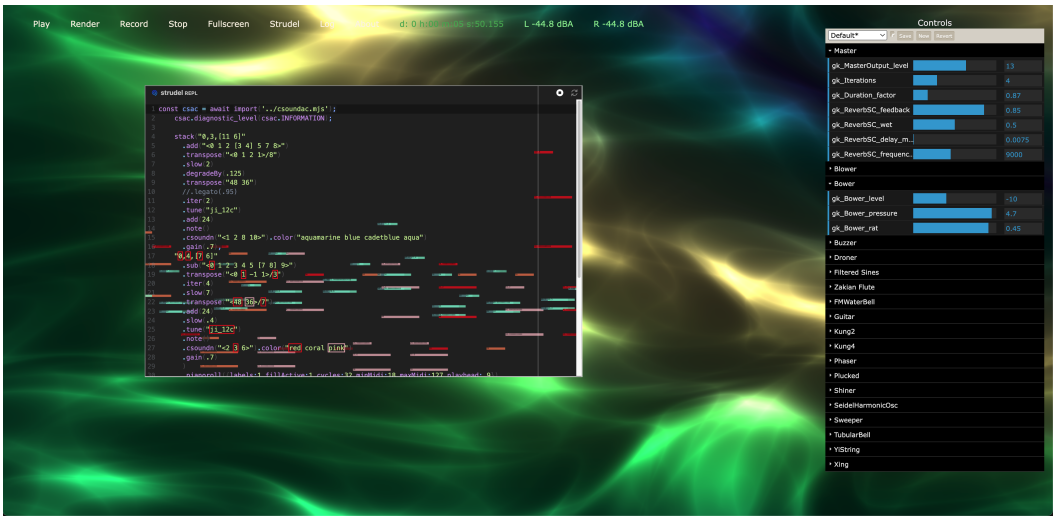


Fig. 1. cloud-5 Piece with Strudel and Audio Visualization

details of HTML or JavaScript to compose cloud-5 pieces. The user includes these custom elements in their HTML code like any other HTML element, adds user-defined code such as a Csound orchestra or Strudel patch, and assembles the parts using a little JavaScript. Code folding regions make it easy to organize the code and see only the part that is being edited. Fields of custom elements ending in `_overlay` denote references to other custom elements; fields ending in `_addon` denote functions or data that the user must or may define, including code (JavaScript, Csound, Strudel, GLSL) and/or parameters.

`<cloud5-piece>` Defines the main menu of the piece, instantiates Csound and/or Strudel as required, starts and stops performances, hosts a controls menu, and defines some helper JavaScript code.

`<cloud5-piece>.csound_code_addon` A JavaScript string literal containing user-defined source code for a Csound .csd file (which can be quite large).

`<cloud5-piece>.control_parameters_addon` A user-defined JavaScript object whose fields have the names and initial values of Csound control channels.

`<cloud5-piece>.menu_folder_addon` The user calls this with the name of a new folder to be added to the controls menu of the piece.

`<cloud5-piece>.menu_slider_addon` The user calls this to create a new new slider control in a folder, specifying its Csound channel name, lowest value, and highest value.

`<cloud5-piece>.score_generator_function_addon` A user-defined JavaScript function that will be called at the start of performance to generate a CsoundAC score for performance by Csound.

`<cloud5-piano-roll>` An overlay that draws a three-dimensional piano roll display of a generated score, showing the current play position with a moving ball.

`<cloud5-strudel>` A popup IFrame that shows the Strudel REPL, in which the user can do live coding of the Strudel patch during performance. The Strudel REPL has its own real-time piano roll display, and highlights the currently active functions in the Strudel code.

`<cloud5-strudel>.strudel_code_addon` A JavaScript string literal containing a user-defined Strudel patch to perform the piece.

`<cloud5-shadertoy>` An overlay that shows a canvas displaying a GLSL shader. This shader can be used to visualize audio, to sonify animations, and so on. The element is designed to support the easy adaptation of shaders developed in the ShaderToy Web site [27].

`<cloud5-shadertoy>.shader_parameters_addon` A user-defined Javascript object with the following fields:

- `fragment_shader_code_addon` A JavaScript string literal containing user-defined GLSL code to be compiled for display on the canvas of the shader overlay.
- `vertex_shader_code_addon` May contain user-defined GLSL code to be compiled for display on the canvas of the shader overlay; there is a default value that includes the entire canvas.

`pre_draw_frame_function_addon` May be set to user-defined code in the form of a JavaScript function that will be called just before drawing each shader animation frame, e.g. to set uniforms that control the shader; commonly used to implement an audio visualizer.

`post_draw_frame_function_addon` This may be set to user-defined code in the form of a JavaScript function that will be called just after drawing each shader animation frame, e.g. to sonify animations by translating them to Csound notes.

`<cloud5-log>` An overlay that presents a scrolling list of runtime messages from Csound.

`<cloud5-about>` An overlay showing license, authorship, credits, program notes, and so on.

4 Assembling a Piece

cloud-5 is not a program, it is a toolkit for constructing pieces that are programs. All cloud-5 pieces require the user to include predefined custom elements and to assemble them along with with user-defined addons. The following outline shows how the components of a piece may be assembled:

```
<script>
    let cloud5_piece = document.querySelector("cloud5-piece");
    cloud5_piece.csound_code_addon = document.querySelector("#csd").textContent;
    cloud5_piece.score_generator_function_addon = async function () {
      // User-defined source code here.
    };
    cloud5_piece.strudel_overlay = document.querySelector("cloud5-strudel");
    cloud5_piece.strudel_overlay.strudel_code_addon =
        document.querySelector("#strudel-code").textContent;
    cloud5_piece.control_parameters_addon = {
        "gk_MasterOutput_level": -7,
    };
    let Master = cloud5_piece.menu_folder_addon("Master");
    cloud5_piece.menu_slider_addon(Master, "gk_MasterOutput_level", -50, 50);
    let cloud5_piano_roll = document.querySelector("cloud5-piano-roll");
    cloud5_piece.piano_roll_overlay = cloud5_piano_roll;
    let fragment_shader = document.getElementById("draw-shader-fs").textContent;
    cloud5_shader.shader_parameters_addon = {
        fragment_shader_code_addon: fragment_shader,
    };
    cloud5_piece.shader_overlay = cloud5_shader;
    let cloud5_log = document.querySelector("cloud5-log");
    cloud5_piece.log_overlay = cloud5_log;
    let cloud5_about = document.querySelector("cloud5-about");
    cloud5_piece.about_overlay = cloud5_about;
</script>
```

5 Best Practices

The cloud-5 system is designed for creating permanent works of music – pieces that will always play into the far future (assuming that Web standards continue to be versionless and backwards-compatible, as they have been for 35 years). cloud-5 is not at all a general purpose Web development system, and therefore pieces should not be developed in the standard way.

- Use only local, static resources (e.g., do not use content distribution networks, but rather download all required scripts, etc., to the Web directory). This ensures that pieces will never break due to missing links to external resources.
- Use no tooling (e.g. no rollups); edit pieces directly in the Web directory. This ensures that pieces will never break due to tooling changes, and will be easy to debug; also, that new pieces may be developed in the Web directory without need of a build system.
- As far as possible, keep all components and resources of a piece in one HTML file, e.g. embed Csound orchestras and Strudel patches in the HTML code.

6 Conclusion

Live examples of cloud-5 pieces may be found at https://gogins.github.io/, source code and binary releases may be found at https://github.com/gogins/cloud-5.

To write new compositions: Download the release archive, unpack it somewhere, run a local Web server there, and write new pieces there as HTML pages using any code editor.

References

1. Isacson, Walter: The Innovators: How a Group of Hackers, Geniuses, and Geeks Created the Digital Revolution. New York: Simon and Schuster (2015).
2. Hafner, Katie and Matthew Lyon: Where Wizards Stay Up Late: The Origins of the Internet. New York: Simon and Schuster (1998).
3. Lessig, Lawrence: Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control. City of Westminser: Penguin Books (2004).
4. Mandiber, Michael: The Social Media Reader. New York: NYU Press (2012).
5. Zuboff, Shoshona: The Age of Surveillance Capitalism. New York: Public Affairs (2019).
6. Internet Engineering Task Force: I E T F https://www.ietf.org Accessed 23 March 2024).
7. W3C: Making the Web work. https://www.w3.org (Accessed 23 March 2024).
8. Ecma: Emca International. https://ecma-international.org (Accessed 23 March 2024).
9. HTML 5 Test: https://html5test.co (Accessed 23 March 2024).
10. Jangda, Abhinav, Bobby Powers, Emery D. Berger, and Arjun Guha. Not So Fast: Analyzing the Performance of WebAssembly vs. Native Code. https://arxiv.org/abs/1901.09056 (2019).
11. Awesome WebAssembly Languages https://github.com/appcypher/awesome-wasm-langs (Accessed 10 July 2024).
12. W3C: Web Audio API. https://webaudio.github.io/web-audio-api (11 March 2024).
13. Katz, Robert A.. Mastering Audio: The Art and the Science, Third Edition. Netherlands: Focal Press (2015).
14. Bassal, Dominique: The Practice of Mastering in Electroacoustics. https://cec.sonus.ca/pdf/The_Practice_of_Mastering.pdf (December 2002).
15. Bainter, Alex: web. music. generative art. https://alexbainter.com (Accessed 23 March 2024).
16. Primozic, Casey: Web Synth. https://synth.ameo.dev (Accessed 24 March 2024).
17. Roberts, Charlie: Gibber https://gibber.cc Accessed 24 March 2024).
18. Roos, Felix, Alex McLean, et al.: Strudel REPL. https://strudel.cc/ (Accessed 23 March 2024).
19. cloud-music: Computer Music on the Web https://AuthorA.github.io (Accessed 23 March 2024).
20. Gogins, Michael: csound-wasm https://github.com/gogins/csound-wasm (Accessed 11 July 2024).
21. Csound Github site, http://csound.github.io.
22. lazzarini, V. et al.: Csound: A Sound and Music Computing System. Springer (2016)
23. Csound Community: The Canonical Csound Reference Manual, Version 6.18.0 https://csound.com/docs/manual/index.html (Accessed 23 March 2024).
24. Csound Developers: Csound API 6.18. https://csound.com/docs/api/index.html (Accessed 23 March 2024).
25. Gogins, Michael: csound-ac https://github.com/gogins/csound-ac (Accessed 11 July 2024).
26. Gogins, Michael: Csound AC 1.0.0 https://github.com/gogins/csound-ac/blob/master/csound-ac.pdf (Accessed 11 July 2024).
27. Quilez, Inigo, Pol Jeremias, et al.: ShaderToy BETA https://www.shadertoy.com (Accessed 24 March 2024).

GUIs and skills in Live-electronics

Cabbage is dead, long live Cabbage!

Rory Walsh *

Dundalk Institute of Technology
rory.walsh@dkit.ie

Abstract. In April of this year, JUCE announced a new end-user license agreement. While the updated license doesn't signify the immediate demise of Cabbage in its current form, it has presented a unique opportunity to reassess the project as a whole. Consequently, a new version of Cabbage is currently under development from the ground up. The end-user experience will remain largely unchanged: the familiar Cabbage syntax will persist, users will retain access to a wide array of widgets, and they will still be able to export to all popular plugin formats. However, the bulk of the new work will occur behind the scenes. This redesigned version will feature a significantly reduced codebase. Moreover, it will leverage the power of VS Code, providing developers with more options to create modern, responsive, and dynamic user interfaces.

Keywords: Csound, Cabbage, Audio Plug-Ins, JUCE, VST, AU

1 Introduction

Cabbage is a framework for developing audio software using Csound[1]. It was initially released in 2008 at the Linux Audio Developers conference [2]. This first version provided users with a way to create cross-platform standalone audio software with a customized UI. In 2011, a new version of Cabbage of released[3]. This made the transition from wxWidgets [4] to JUCE [5], which enabled the integration of two earlier projects, namely csVST and csLADSPA [7]. Moreover, it facilitated the export of plugins as AudioUnits. Since its inception, the primary goal has always been to help users leverage the capabilities of Csound within a DAW.

2 Cabbage and JUCE

Since 2011, Cabbage has extensively utilized JUCE, the world's most popular audio application and plugin development framework. JUCE has always offered a dual license: you can release software under GPL or purchase a commercial license for closed-source projects. As there was no necessity for a commercial license, Cabbage has consistently been released under the GPL. It also ships with over a collection of over 300+ instrument contributed by Iain McCurdy and released under the CC Share Alike 4.0 International license.

In 2016, users on the Cabbage forum began inquiring about releasing commercial products with Cabbage. While the GPL license doesn't prohibit this, users would need to distribute their Csound source code as well. For various reasons, some users felt uncomfortable with this requirement and privately approached me about a commercial version of Cabbage. In 2018, Puremagnetik released the first commercial plugin using Cabbage.

2.1 Cabbage Pro

From this initial commercial release, the so-called "pro" version of Cabbage emerged. To enable this, a commercial license for JUCE had to be acquired. The sole distinction between the 'pro' and public versions is that the pro version encrypts the Csound source files; all other aspects of the software remain identical. To date, Puremagnetik has released over 50 Csound-based audio plugins, receiving rave reviews[9].

* I'm indebted to all Csound users and developers. Without their input and support, Cabbage simply wouldn't exist.

2.2 High-Profile Cabbage projects

In 2022, Kia released a software instrument[10] developed in collaboration with DaHouse Audio, Brazilian synth maker Arthur Joly, and AudioFB. Raphael Gomez at AudioFB wrote the Csound code for this instrument. This collaborative effort resulted in the creation of the Kia instrument, which draws inspiration from Joly's analog synthesizers.

In May of this year, Coca-Cola released Coke SoundZ. This instrument was the result of another collaboration between DaHouse and AudioFB. It provides users with the tools to create melodies from various sonic entities of a Coke bottle, such as 'phst,' 'fizz,' 'clink,' 'glug,' and 'ahh.' By pressing the AI Generate button, users can create unique sounds based on thousands of audio samples. Additionally, the instrument was port to a physical device: a Coke bottle that emulates the audio software using tactile pads, sliders, switches, and buttons. Although Cabbage is not used in the bottle, Csound is still at the heart of this device and runs on an a RPi Zero w2 which is housed within the device.

2.3 JUCE 8

JUCE 8 is set to be released in 2024. A draft of the new end-user agreement was shared on the JUCE forum in April [**?**]. Within days, the thread was inundated with over 500 messages from audio developers expressing concerns about the potential negative impact on their work. The most significant proposed change was that anyone contributing to a JUCE-based project would require a license, regardless of whether their contribution directly utilised JUCE. This meant that artists providing presets, graphic designers contributing visual assets, and developers contributing framework-agnostic code would all need a license under the terms of the initial draft.

From a Cabbage perspective, the major concern arose from the requirement for users to obtain a JUCE license if they create presets. Every Cabbage instrument essentially functions as one extensive preset. However, this primarily affects only about a dozen Cabbage users who utilize the pro version. Other notable changes include a re-licensing of a substantial amount of ISC code and a transition from GPL to AGPL.

3 Cabbage without JUCE?

As mentioned earlier, the revisions to the JUCE EULA don't signify the end for Cabbage, but they highlight the system's vulnerability due to its reliance on a commercial framework. Future changes to the EULA could pose further challenges for the project, and if the GPL license option were to change, Cabbage would find itself in an extremely precarious situation, given that almost all of its codebase utilizes JUCE classes. Replacing this code is a huge undertaking, but it might also present an opportunity to reassess the project and scale back on over a decade of feature creep.

When Cabbage was first released, JUCE was the only choice when it came to audio frameworks. However, the landscape has since improved. The DISTRHO Plugin Framework[12] provides a comprehensive collection of wrappers for the most common audio formats, as does CPlug[13], a C-based plugin interface. Among the current plugin frameworks in development, iPlug [14] is arguably the most mature. Developed and maintained by Oli Larkin, it offers excellent support for a range of plugin targets. Although these frameworks may not match the functionality of JUCE, they are all released under permissive licenses. As of the time of writing, iPlug2 appears to be the best fit for Cabbage version 3.

4 Cabbage, a complete redesign

Cabbage 3 represents a complete rebuild of the current project and presents the opportunity to remove some of the more maintenance heavy elements of the software and replace them with more sustainable implementations. Outlined in the following sections are the places that will benefit most from this rewrite.

4.1 Code editor

The first part of Cabbage to undergo significant changes is the IDE itself, in particular, the code editor. Originally built using the JUCE code editor component, it lacked essential functionality such as auto-indentation, column edit, and line highlighting, all of which had to be implemented manually. Adding these editing features felt disconnected from the core focus of audio programming. Despite being relatively functional, the code editor always felt clunky. Instead of attempting to rebuild it using another framework, the decision was made to switch to VS Code.

Initially, the plan was to integrate Monaco, the VS Code editor component, within a webview in Cabbage. While a basic proof-of-concept showed promise, users wouldn't be able to fully leverage the power of VS Code when developing Cabbage plugins. Consequently, a native VS Code extension was developed, providing commands for exporting and running instruments directly from within VS Code. The syntax highlighting is provided by Steven Yi's csound-vscode language extensions[8]. On top of this, all the audio/MIDI settings can now be modified and updated from the extension settings page.

4.2 User interface designer

The UI designer in Cabbage was expected to be a relatively straightforward feature to implement but turned out to be one of the most complex areas of the codebase, primarily due to its integration with the code editor. The new UI designer has been developed using the VS Code Extension API and is now launched within a webview inside VS Code. This transition from handling and updating component layouts and properties in C++ to a much simpler solution using HTML/CSS/JS has significantly reduced complexity.

In fact, the re-implementation of the UI designer, alongside the Cabbage syntax parser, was completed in just 3 days. This overhaul allowed for the removal of over 6000 lines of C++ code [1]

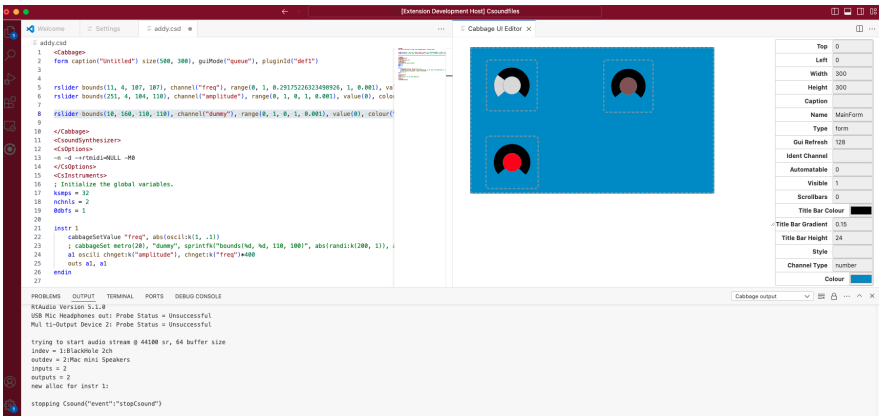


Fig. 1. The new UI designer with property panel, and Csound output console integrated within VS Code

4.3 The Cabbage plugin UI

The component library shipped with Cabbage constitutes approximately one-third of the entire project's source code, with the slider classes alone comprising over a thousand lines of code. Given that plugins themselves are largely widget-agnostic, this amount of code appears somewhat excessive, particularly considering that users typically utilize only a handful of widgets. The opportunity to eliminate this code provided far too appealing and will ultimately relieve a significant maintenance burden.

[1] These figures are somewhat anecdotal as the C++ code in question could probably be refactored and reduced.

4.4 Embedded webview

The UI designer in the new Cabbage VS Code extension utilizes a webview and HTML/CSS to display user interfaces. Given the effectiveness of this approach, the decision to migrate the entire UI to a webview hosted by the plugin was promptly made. However, this migration comes with some caveats. On one hand, communication between the UI and plugin may be slightly slower, as messages need to be passed from the plugin to the webview. Although the plugin can directly invoke JS code, messages must be transmitted as strings. Despite this slight inconvenience, early indications suggest that there is no significant impact on performance. On the other hand, graphics rendering is now much faster. Additionally, this approach offers the benefit of allowing users to develop entirely customized UIs in HTML/CSS, including high-resolution visualizations, which was something the old version of Cabbage struggled with.

5 How Cabbage and VS Code work together

The current version of Cabbage launches a standalone plugin when users compile an instrument. This approach was considered in this new design, but ultimately felt a little awkward to constantly have to move from VS Code to another program when developing instruments. Instead, the interface is embedded into a VS Code panel. When a user saves/compiles an instrument, a headless version of Cabbage is launched that compiles the current .csd file. A web-socket connection is set up between the UI in VS Code and the headless Cabbage application running in the background. It's important to note that while providing users with a far more powerful means to create UIs, Cabbage 3 will ship with the same widgets as the current version. Most have already been ported to HTML/CSS/JS.

6 Conclusion

Rebuilding Cabbage from the ground up is no small feat, but it presents significant opportunities to streamline bloated code and remove rarely used features that have accumulated over the years. Transitioning from JUCE to a permissively licensed framework will enhance the project's longevity and sustainability. Additionally, all communication protocols and interoperability have been implemented using permissive libraries, further ensuring the project's future-proofing.

While some users may not like the switch from an in-built code editor to using VS Code for instrument development, it's essential to recognise its superiority over the current editor. Currently, Cabbage 3 has progressed beyond the proof-of-concept stage, and the results are very promising. It is hoped that a public alpha version will be available for user testing within the next few months.

References

1. Lazzarini, Yi, ffitch, Heintz, Brandtsegg, McCurdy, "Csound: A Sound and Music Computing System". Springer; 1st ed. 2016 edition (21 Nov. 2016)
2. Walsh, R. "Cabbage, a new GUI framework for Csound". Proceedings of the Linux Audio Developers Conference KHM Cologne, Germany. 2008
3. Walsh, R "Cabbage Audio Plugin Framework." Proceedings of the International Computer Music Conference, Huddersfield. 2011
4. wxWidget Homepage, `https://www.wxwidgets.org/`
5. JUCE Homepage, `https://www.juce.com/`
6. Lazzarini, Walsh, Brogan "Two Cross-Platform Csound-Based Plugin Generators". Proceedings of the International Computer Music Conference, Belfast. 2008
7. Lazzarini, Walsh "Developing LADSPA plugins with Csound" Proceedings of the Linux Audio Developers Conference TU Berlin, Germany. 2007
8. Csound VSCode Plugin Extension, `https://marketplace.visualstudio.com/items?itemName=kunstmusik.csound-vscode-plugin`
9. Puremagntik Devices `https://puremagnetik.com/collections/devices`
10. Kia Instrument Homepage `https://www.kia.com/us/en/movement/our-instrument`
11. Coke Soundz Homepage `https://www.kia.com/us/en/movement/our-instrument`
12. DISTHRO Plugin Framework Homepage `https://github.com/DISTRHO/DPF`
13. CPLUG Homepage `https://github.com/Tremus/CPLUG`
14. iPlug2 Homepage `https://iplug2.github.io/`

Envelope Shaper GUI for Complex Curves in Csound

Gianni Della Vittoria

Liceo Artistico e Musicale A. Canova di Forlì
gianni.dellavittoria@liceocanovaforli.edu.it

Abstract. Creating envelopes is a valuable resource for giving movement to sound. Here we present a tool that facilitates the creation of complex envelopes thanks to a graphical interface in which the user can quickly draw the curves necessary for the most varied musical purposes. Four typical needs in the creation of the envelope are identified and discussed: the management of the general profile, the tremolo, the loop, the random component. The output product of this software will be Csound code. Designed particularly for beginners who start learning Csound, this tool makes it possible to facilitate the understanding of the envelope in the context of the parameter on which it is applied, and to provide ready-made code useful especially in conditions of very complex shapes.

Keywords: Envelopes, Csound code, Curves, Loop, Random

1 Introduction

One of the many advantages of Csound is that it provides a long series of effective opcodes useful for creating the most diverse envelope shapes. Furthermore, the flexibility of the language allows envelope profiles to be further processed in endless ways.

Here I mean by envelope any form useful for managing the evolution of any parameter over time. We know how precious a resource this is, as the movement that an envelope manages to impart to the various parameters can alone completely change the final effect on the sound.

Musical needs often require complex envelope profiles and defining them in Csound code can sometimes be difficult, especially for a beginner. In the wake of a previous article of mine from 2022 "Educational Tools for Csound", this work is aimed above all at helping those approaching this programming language, simplifying the understanding and creation of complex curves for envelopes.

This article, therefore, has a dual purpose. On the one hand it presents a tool[1] capable of producing the design of envelopes with complex shapes with relative ease, thanks to an intuitive graphic interface. This tool will output the Csound code of the processed form. On the other hand, it wants to propose a more general reflection on how an envelope suitable for various musical needs can be conceived today.

2 Envelope Settings

Often, when searching for the right envelope, you have at least one of these needs:
- describe the general progression of the parameter
- add tremolo effects
- repeat a fragment in a loop

[1]The Envelope Shaper GUI for Csound is written in Python and is available on GitHub.

- contemplate a random component

These four categories could be used individually, but also combined in various ways. For example, you might want to make a loop but each time it repeats it moves higher. Or a tremolo that gradually becomes irregular thanks to a random component.
The basic idea is to provide a graph in which to show the four types of envelope that can be processed, which will then simply be added together.
These types are respectively:
- One-shot
- LFO
- Loop
- Noise

The user can control the contribution of each of them and see the final result both visually and in Csound code. Thus, to obtain the envelopes of the previous example, in the first case it will be sufficient to use Loop and One-shot, in the second LFO and Noise.

3 One-shot

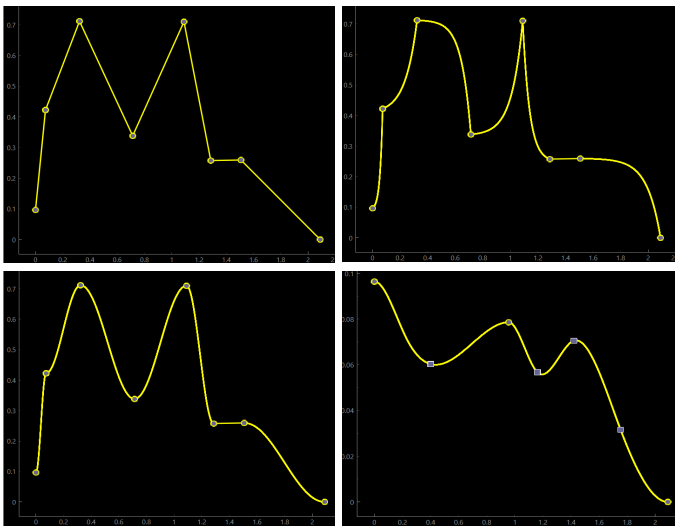
This is the form that does not require any repetition, but consists of a single reading from left to right.

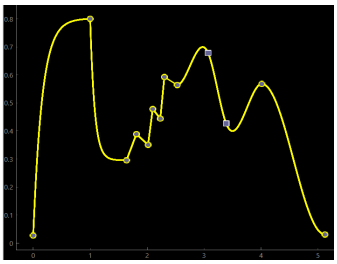
It is made of a series of segments delimited by handles defined by the user by clicking and dragging the vertices. On the x-axis you have the time in seconds, on the y-axis the expected range.

3.1 Possible Forms for Segment

Each segment between two handles can have its own shape, independent of that of the other segments. The most typical shapes are available, including:

- linear: the vertices are connected by a straight line (*linseg*)
- exponential with curvature index: the segments can have a more or less pronounced convexity, definable by vertical dragging between the two segment handles (*transeg*)
- sigmoid: the curve smoothes the transition from one vertex to another (*cosseg*)
- cubic splines: the connection passes between further user-defined secondary points within the segment as smoothly as possible, guaranteed by the cubic interpolation algorithm (*Gen 08*)





The richness of the final design will benefit from the possibility of adding different shapes to each new segment.

3.2 Absolute and Relative Timing

Since the system is mainly aimed at execution in no real-time, with the possibility of instantiating an instr several times with different durations, it is important to be able to choose between an execution with absolute timing, i.e. defined in real seconds, and an execution with timing relating to the duration of the instance itself (p3). For example, if an envelope in two segments rises from 0 to 1 in 2 relative seconds and returns to 0 in 3 relative seconds, it means that it will make the first upward ramp taking 2/5 of the total duration p3, and 3/5 of p3 for the following descent, whatever the duration of p3.
Since it is possible to mix segments of absolute and relative duration, graphically distinguishable by the dashed line in the case of relative duration, the relative duration is intended to be proportional to the duration of the instance once the absolute part has been subtracted (p3 - sum of the absolute parts).
Thanks to this system it is simple to reproduce the typical ADSR case as well as its different variants. In fact, it will be sufficient to place the sustain segment in dashed lines to indicate relative duration.

3.3 Random point position for each handle

To have slightly different envelopes at each new instance of the Csound instrument it is possible to declare the degree of randomness of the position of the individual handles. Each handle can have a vertical line oriented downwards of the length chosen by the user indicating the range within which the real value will be randomly taken.

Similarly, each handle apart from the first can have a horizontal line oriented towards the left indicating the time frame within which the temporal position of the handle will be randomly chosen. This section, which can also be extended at the user's discretion, cannot go beyond the position of the previous handle.

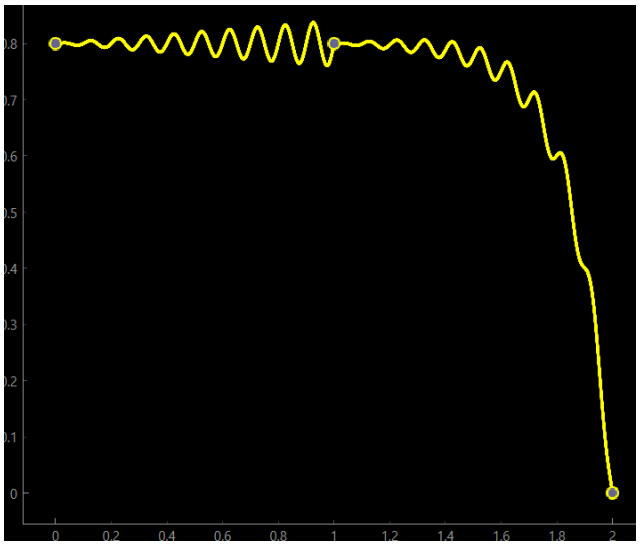
In essence, for each handle a probability rectangle can be drawn proportional to the value and position of the handle itself, ensuring the desired degree of flexibility without betraying the original shape.

4 LFO

It is a low-frequency bipolar oscillator based on classic preset shapes, typically used to add tremolo and vibrato effects.

Alongside the choice of the waveform and the definition of the initial phase, two parameters are particularly important: amplitude (depth of the vibrato) and frequency (speed of the vibrato).

Both can undergo variations during the execution of the envelope, therefore both the amplitude and the frequency are defined by their respective dedicated one-shot envelopes. So, for example, you can start a note without vibrato and then gradually introduce the effect by increasing its depth.



5 Loop

By Loop we mean an envelope with the characteristics of the one-shot envelope, which however will be repeated entirely in a loop until the duration of the instance is exhausted. This is the kind of envelope that in Csound is defined by the *looptseg* opcode.
The Loop window has 3 envelopes:

- Amplitude
- Frequency
- Loop shape

Useful for example to make a step sequencer, or simply to draw a custom-shaped LFO. Here too it is possible to determine amplitude and frequency by respective one-shot envelopes, in order to manage the loop by varying its depth and speed along the execution of the instance.
There are also more specific tools to facilitate use for a step sequencer.

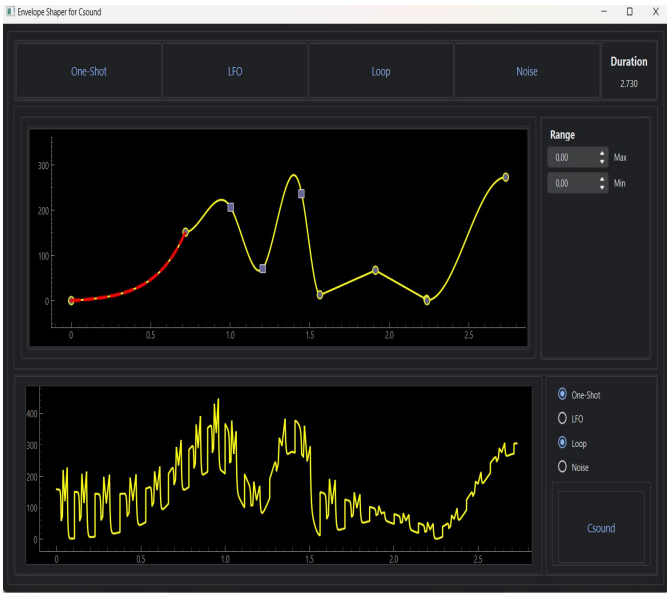
6 Noise

This is a bipolar signal obtained by creating linearly interpolated random numbers. What you get on Csound with the *randi* opcode.
Here too it is possible to manage amplitude and frequency independently, as usual by dedicated one-shot envelopes.
It's useful for "dirtying" the envelope a bit, for example. Or to obtain a very random trend, if the amplitude of Noise prevails over that of the other types of envelope that are added.

7 Algorithmic variations

In envelopes with a fair number of handles, it can be tedious to make changes by acting on each of them one by one. It is then advisable to use algorithms specifically designed to carry out typical operations, such as:

- vertical stretching of the handles
- temporal stretching
- vertical offset
- clipping
- smoothing to round off the edges
- simultaneous change of curvature indices
- points simplifier
- presets management



8 Conclusion

Understanding the nature and purpose of an envelope, which certainly cannot ignore the context, in particular the type of parameter on which it is applied, remains a fundamental stage in the production of electroacoustic music, especially for Csound beginners. The intrinsic possibilities that Csound offers certainly go beyond what was said above, but the ability to concretely see the shape of the envelope during its conception, to make changes quickly, to save and recover presets can pave the way for a more conscious use of the most complex shapes without getting lost in details.

References

1. Della Vittoria, G.: Educational Tools for Csound.
https://csound.com/icsc2022/proceedings/Educational%20Tools%20for%20Csound.pdf
2. Lazzarini, V. et al.: Csound: A Sound and Music Computing System. Springer (2016)
3. Csound Github site, http://csound.github.io

Cordelia, crafting a method while live coding in Csound

Jacopo Greco d'Alceo

jacopo.grecodalceo@gmail.com

Abstract. This paper introduces *Cordelia*, a domain-specific language offering an intersection between live coding and contemporary composition practices. Designed to generate Csound and other code on demand, *Cordelia* integrates diverse musical elements such as envelopes, tuning systems, and various instrument types. By prioritising resource efficiency and flexibility, it enables seamless transitions between live coding session, DAW scripting in environments like Reaper, and graphical scoring. The paper highlights some particularly features and *Cordelia*'s architecture, suggesting its potential to broaden the creative possibilities of contemporary composition.

Keywords: composition, method, live coding, Csound, contemporary music

King Lear: Speak.
Cordelia: Nothing.

W. Shakespeare

1 Introduction

Cordelia is born as part of a personal artistic research focusing on developing a methodology and creating an environment for contemporary composition. It is a *domain-specific language* built with Python, mainly adapted for exploiting the potential of Csound code. This abstract and personal language with his technical propositions and his artistic intentions embed inside its intimate pre-composed structure, becomes the intermediary between the musical and compositional thought and its concrete, temporal and dramaturgical result. The work *chiedimi le mie radici*[1] is a direct example of this conjunction and ideas.

One of the main features of *Cordelia* is its ability to respond to current artistic needs by *generating code only when it is required*. For instance, Csound instruments are compiled only *in the very moment* the instrument is used during each session. This approach is extended to all elements external to the main core, such as tuning systems, audio files, and opcodes. Everything is deferred until the moment of use, enabling an extensive library of various file formats to be available on-the-fly without overloading memory. This reflects a philosophy of resource efficiency without compromising wide-range functionality, which is emblematic of live coding. *Cordelia* was in fact originally conceived as a *live coding language* due to its close resemblance to the compositional process; it has since evolved into a versatile tool that navigates the ambivalence between *algorithmic improvisation* (e.g. Csound, Python) and *horizontal timeline editing composition* (e.g. Reaper), while also integrating *graphical elements of notation* (e.g. Lilypond, Abjad).

2 Architecture

In order to maintain a flexible environment, *Cordelia*'s architecture is structured with an inner core primarily managed with Python and an outer shell that includes different format.

Each external component is preserved in its standard format, as closely as possible to its original form (i.e. tuning systems in `.scl`, audio files in `.wav`, tables in `.orc`). The goal is to enable the quick addition or removal of files while ensuring that results remain accessible in other environments. All

```
Cordelia
├─SHELL
│  ├─GENs - envelopes tables and waveforms (.orc)
│  ├─INSTRs - instruments
│  │  ├─instr - Csound code instruments (.orc)
│  │  ├─sonvs - Standard sound file recording (.wav)
│  │  └─hybrid - Hybrid instruments: .orc using sound files recording
│  ├─MODs - sound effects as UDOs (.orc)
│  └─SCALA - tuning systems (.scl)
└─CORE
```

Fig. 1. Cordelia's main structure

these files are converted as needed into Csound-ready formats by the language. The primary approach involves analysing the files and storing them in a `JSON` format, where they are converted into essential code ready for use by Csound.

The core system launches an instance of Csound through the `ctcsound Python API` and waits for messages via a `UDP` server. These messages are parsed to generate instrument code or score instructions. For example, messages can be sent from a text editor (e.g. Vim or Neovim) or from a DAW (e.g. Reaper) for live coding or single-event composition.

Csound allows to directly replace an instrument in order to update any relevant parameters. As demonstrated in the following section, a multiple-variable approach is preferred, skipping initialisation phase, enhancing fluidity and continuity in the music while taking advantage of the Python code layer.

3 *Cordelia*'s code structure

The code structure is inherited from Csound, meaning that each parameter has its *own fixed position* within a block of code (or *node*). The specific position of each parameter can vary depending on the node structure. The following code represents an abstract model of one of the primary rhythmic structures:

```
rhythmic function: p1, p2, p3, p4, p5, [p6], [p7], ...

Parameters:
- p1: Instrument name (type: string, format: @ + name)
- p2: Duration (type: keyword or number, format: seconds, includes release time)
- p3: Dynamic level (type: keyword or number, range: 0 to 1)
- p4: Envelope (type: keyword, generally a ftgen global variable)
- p5: Frequency (type: number, unit: Hz)
- p6 [optional]: Additional frequency (type: number, unit: Hz)

Note:
- p0 [optional]: Space (type: number, each number is a specific speaker)
```

The space parameter serves two purposes: it allows independent management of the instrument relative to the number of speakers (each instrument is "mono", with an individual instance for each speaker), and it controls where the sound is routed. If omitted (`p0 = 0`), the sound is directed to all speakers. Unlike traditional systems that create a single instance of an instrument and direct it to all outputs, *Cordelia* creates *as many instances as there are speakers*, treating each source as a *separate entity*. This channel management approach allows users to write variations for each channel and control spatial placement intuitively. With this method, sound can be distributed in geometric sequences across multiple points, enabling *Cordelia* to orchestrate multichannel spatialisation through what can be described as a hard-panning coding setup. Future developments will focus on integrating additional spatialisation systems to further enhance its capabilities.

Additional sound effects, provided as opcode, can be added after the name with a simple dot to separate them (i.e. `@aaron.delay(1)`).

4 An example

```
saf: "-u-u", 4  ; saf is an opcode generating rhythm inspired by Greek prosody
    @aaron      ; the name of the instrument: "aaron"
    qn          ; "qn" stands for "quarter note"
    mf          ; "mf" stands for "mezzo forte"
    iago        ; "iago" stands for "giiago", a name for a ftgen envelope table
    300         ; 300 means 300Hz
```

This code is converted into the following steps.

At the beginning of a live coding session, in the very moment some code is sent to *Cordelia*, all the instruments, files, tables, modes, or scales that are currently used are converted, prepared and finally sent to Csound. Thus, they are stored in a Python list that serves as memory of the already used component in the session. Every time an instrument is called, immediately *Cordelia* creates a named array in order to store and route the instrument's output through the Csound `chnset` and `chnget` system (and consequently, a `chnclear` instrument is started at the end of the chain - after the main outputs).

Following the example, the code generated will be:

- `gSaaron[] init ginchnls`: a string array that stores the named audio outputs is declared based on the number of output channels.
- `gSaaron[0] sprintf "aaron_%i", 1` and `gSaaron[1] sprintf "aaron_%i", 2`: in a stereo situation, the previous array is now specifically filled with the instrument name linked string for each channel. Consequently inside each Csound instrument there is `chnmix` that directly connects with this specific string.
- `schedule 950.0003, 0, -1, "aaron_1"` and `schedule 950.0004, 0, -1, "aaron_2"`: this will increase and turn on a `chnclear` instrument `950` that will clear the `chnmix` outputs of the recently created instrument.

The instrument that stores the main tempo and the instrument managing the final main outputs are activated during the lunch of *Cordelia*.

After this initialisation phase, a unique global variable associated with the source instrument name is created for each parameter. Each of these variables corresponds to its own instrument.

```
    instr 215
gSname_aaron1 = "aaron"
    endin
turnoff2_i 215, 0, 1
schedule 215, ksmps / sr, -1

    instr 216
gkrhythm_aaron1 saf "-u-u", 4
...
...
```

These instruments numbers are stored in a Python list and linked with the parsed node. They remain unaffected until the node is erased — in which case, a `turnoff2_i` command is sent to all of them. If a parameter changes, they are compared with the previous state and eventually replaced and updated.
Each of these global variables converges into the following subsequent instrument responsible for the rhythmic pattern (typically structured with an `if` statement).

```
    instr 222
if gkrhythm_aaron1 != 0 then
    eva(gkspace_aaron1,
        gSname_aaron1,
        gkdur_aaron1,
        gkdyn_aaron1,
        gkenv_aaron1,
        gkfreq1_aaron1
    )
endif
    endin
turnoff2_i 222, 0, 1
schedule 222, ksmps / sr, -1
```

The `eva` opcode is a simple schedule instrument generator `UDO`.

This concludes the control and patching section. Finally, to route the sound to the sources, a last instrument is created connecting the array mentioned above to the *Cordelia* main instrument output and eventually parsing the effects.

This structure allows for changing a single parameter without affecting the others.

5 Additional projects

Four additional projects are closely tied to *Cordelia*'s architecture and will soon be integrated into its core. *Cordelia-ghost*[1]: it produces high-quality spectrograms ready to be printed on score or vice versa. Again the idea is to generate hybrid graphical score directly after a live coding session or a Reaper project. *Cordelia-cue*[2]: an excerpt of the core of *Cordelia* designed to operate in conjunction with a specific `.csd`, facilitating the execution of a cue list during live performances. This lightweight program boasts efficient memory usage, ensuring optimal performance during instrumental scores that embed electronic. *Cordelia-csound*[3]: an interface seamlessly integrated into Reaper allows sound to be directly manipulated inside Reaper during editing merging the full potential of *Cordelia* as a Csound connection to analyse and transform sounds[4]. *Cordea-litchi*[5]: an effort to merge algorithmic generation of Lilypond scores and translate them into corresponding Csound scores.

6 Conclusion

This article offers a glimpse of *Cordelia* as a dynamic instrument for contemporary composition. Many reflections come with the creation of a language and a lot of arguments still has to be treated. *Cordelia* remains open to discussion, suggestions, and exchanges, welcoming collaboration and input from the community. As *Cordelia* evolves in its future development, a focus on comprehensive documentation will ensure accessibility and usability for users of all levels. Moreover, *Cordelia* aims to reestablish Csound as a premier language for live coding, focusing on its power and flexibility to push the boundaries of musical expression and its incredible score. Gratitude is extended to Steven Yi's work on live coding and the Csound community for their invaluable assistance and support.

For more information, please visit the project at `https://github.com/jacopogrecodalceo/CORDELIA`.

References

1. Jacopo Greco d'Alceo. chiedimi le mie radici, . URL `https://jacopogrecodalceo.github.io/en/projects/chiedimi-le-mie-radici`.
2. Thor Magnusson. *Sonic Writing: technologies of material, symbolic, and signal inscriptions*. Bloomsbury Academic. ISBN 978-1-5013-1388-2 978-1-5013-1385-1 978-1-5013-1386-8 978-1-5013-1387-5.
3. Steven Yi. Live coding with csound. URL `https://github.com/kunstmusik/csound-live-code/tree/main`.

[1] `https://github.com/jacopogrecodalceo/cordelia-ghost`
[2] `https://github.com/jacopogrecodalceo/cordelia-cue`
[3] `https://github.com/jacopogrecodalceo/CORDELIA/tree/main/rpr/cordelia_csound`
[4] e.g. Analysis - Transformation - Synthesis (ATS) or Linear Predictive Coding (LPC).
[5] `https://github.com/jacopogrecodalceo/cordelia-litchi`

4. Jacopo Greco d'Alceo. Hybridation, a mind-body problem: live coding. . URL `https://forum.ircam.fr/article/detail/biographies-abstracts/#Greco`.
5. Jacopo Greco d'Alceo. International conference on live coding - LiveCoding as life: Cordelia, the utopian tragedy of crafting a method. . URL `https://www.youtube.com/watch?v=QIxhGh35Bbc&t=1639s`.

Csound Live Coding with Multiple Clients

SeoKyeong Toby Kim *

National University in Maynooth
seop0504@gmail.com

Abstract. This paper introduces a Python-based TCP socket server designed for collaborative live coding sessions utilizing the Csound engine, aimed at enhancing group music creation. The server facilitates real-time, multi-client connectivity, allowing users to dynamically create and manipulate custom Csound instruments. This system is equipped with an internal loop mechanism that manages quantized events and chord transitions, providing a rhythmic backbone for musical compositions.
Designed as a fun and innovative project, this server is an excellent platform for both novice and experienced musicians to experiment with collaborative composition and live performance in a digital setting. It provides a playful yet robust framework for musical exploration and interaction.

Keywords: Csound, Python, TCP Socket, Music, Tonal/Modal, Instrument/Effect, Loop, Random

1 Introduction

There are many great Csound live coding projects available like Steven Yi's cosund-live-code application or Victor Lazzarini's litePlay.js. However, this reseaerch wanted to make a more collaborative environment for music creation. Imagine group of people start to make music together in real-time. There could be a team competition. This paper introduces a Python-based TCP socket server utilizing the Csound audio engine to facilitate live coding sessions for collaborative musical performance. The project leverages a server-client architecture, where multiple clients can connect simultaneously, interactively manipulating and creating music.

The server supports dynamic instrument creation and manipulation through a user-friendly command interface, allowing participants to define and control custom instruments and musical loops in real-time. A central feature of the system is its internal timing mechanism, which handles quantized events and chord changes, thus maintaining musical coherence across sessions with multiple participants. This timing system is essential for synchronizing musical events and ensuring that all participants contribute cohesively during live sessions.

Moreover, the server provides a framework for querying musical components such as instruments, loops, and audio buses, thus enhancing the transparency and accessibility of the system. The users can actively engage in musical creation, making adjustments to the loops and events they subscribe to, providing a responsive and interactive musical experience.

The functionality of the server extends to handling complex musical data structures and providing real-time feedback to users, making it an invaluable tool for educational and experimental music platforms. This project aims to explore the potentials of collaborative digital music creation, emphasizing flexibility, ease of use, and real-time interaction, fostering a community of practice among musicians and technologists. This setup not only enhances the musical experience but also promotes an understanding of digital music technologies in a collaborative setting.

2 The System

The system is designed as a TCP socket server, leveraging Python's networking capabilities to facilitate real-time musical interactions via the Csound engine. At its core, the server manages connections from multiple clients, each capable of sending commands that influence the musical output. This setup is built using several key Python modules:

* I would like to thank the professor Lazzarini from National University in Maynooth for insights

2.1 Socket Programming & Threading

Python's socket module forms the backbone, enabling network communications. The server listens for client connections and handles incoming data as commands for musical operations.The threading module is utilized to manage concurrent operations, allowing the server to handle multiple clients simultaneously without interrupting the audio processing. This is crucial for maintaining a fluid and responsive user experience.

2.2 Csound API & Pyaudio

The ctcsound Python interface to Csound allows the server to directly interact with the Csound engine. This integration is pivotal for real-time sound synthesis and musical composition manipulation. Due to Python's limitations in providing a stable clock, implemented PyAudio is implemented within the setup. PyAudio serves as the audio engine that retrieves a stable clock, which is crucial for the precise processing of the Csound engine by ksmps. This setup ensures that the audio processing remains consistent and synchronized, thereby enhancing the reliability and performance of real-time audio operations.

3 Live Coding

Live coding within this system is enabled by a straightforward client-server architecture where the server handles musical data and processing, and clients interact through a network interface. Users connect to the server via a Python client application that allows them to send commands and receive feedback in real-time. Here's how users can engage with the system:

3.1 Client-Side Interaction

The client application "client.py" provides two primary functionalities: sending messages "commands" to the server and receiving messages "responses or updates" from the server. Users input commands which are then packaged and sent over TCP/IP. The communication is handled by separate threads for sending and receiving to ensure that user input and server responses are processed asynchronously, enhancing the interactive experience.

Users type commands into the console, which are read by the `send_messages` function. Each line of input is accumulated until a semicolon (;) is entered, indicating the end of a command.

Example command from client command-line

```
orcIn test
aout poscil p5, cpsmidinn(p4)
aout linen aout, 0.001, p3, p3/3
out aout,aout;
```

See, it does not need instr 'wrapper' like this,

```
instr 1
endin
```

This is because the server will assign a wrapper with instr number to the incoming instrument. User can access to the instrument with the name provided as the first 'arg' after 'orcIn'.

```
scoreIn test 1 60 .5;
```

For a command like cLoopIn,

```
cLoopIn InstrName LoopInterval LoopPosOffset LoopType Durations p4s p5s ...
```

It receives list of loop elements for every p-fields. Each element within the list is separated by comma(,).

```
cLoopIn test 1 0 0 .3,.2 60,64,67 .4,.2;
```

Every p-field loops independently. It means that p3 is going to loop through 0.3 and 0.2 while p4 is going to loop through 60,64,67. By the time when third iteration gets played, p4 is going to be 67 and p3 is going to be 0.3 again.

'LoopType' specifies how the loop behave, 0 means accending, 1 means deccending and 2 means random.

'LoopInterval' and 'LoopPosOffset' is responsive to the clock where '1' means a quarter note. If one wants to play eighth-note, LoopInterval can be 0.5. if one wants to play a quarter note at every beat 3, then it can be expressed like this.

```
cLoopIn test 4 2 0 .3,.2 60,64,67 .4,.2;
```

3.2 Server-Side Processing

On the server side, the Python-based TCP server manages these connections and processes incoming commands to modify the musical environment. Commands might include creating instruments, playing notes, setting loops, and more. Each command is interpreted by the server and translated into actions within the Csound environment, enabling dynamic musical composition and manipulation.

This following code snippet is to register new loop to the system.

```
    orc = f"""
            instr {loopIndex}
            ibpm chnget "bpm"
            \t{getPFields}

            \t{iValDefine}

            \t{triggerStr}
            if {loopControl} == 1 then
            \t{iValAdvance}

            \t{lpStr}
            endif
            endin
            """
        loopControlMap[loopIndex] = classes.LoopControl(loopIndex,lpStr,loopControl)
        cs.compileOrc(f"{loopControl} init 1")
        result = cs.compileOrc(orc)
```

The system recompile new instrument for each loop. It gives flexible way of managing loop components. as mentioned above, every p-field loop independently because every loop defined from the client side will be giArray in csound engine.

```
    for i in range(numP):
            gi_Arrs.append(f"gi_Arr_{giArrayIndex}")
            giArrayIndex+=1
            cs.compileOrc(f"{gi_Arrs[i]}[] fillarray {msg[i+4]}")
            ...
```

Then the newly compiled instrument is looking for the new giarray for loop iteration like this,

```
    instr 1002
            ibpm chnget "bpm"
            iMode = p4
            iField5 = p5
            iField6 = p6
            iField7 = p7

            iVal5 = iMode == 0 ? gi_Arr_0[iField5] : gi_Arr_0[rnd(lenarray(gi_Arr_0))]
            iVal6 = iMode == 0 ? gi_Arr_1[iField6] : gi_Arr_1[rnd(lenarray(gi_Arr_1))]
            iVal7 = iMode == 0 ? gi_Arr_2[iField7] : gi_Arr_2[rnd(lenarray(gi_Arr_2))]

            schedule(2, 0, iVal5*(60/ibpm), iVal6, iVal7)
            if gil1002 == 1 then
                    iField5 = iField5+1 < lenarray(gi_Arr_0) ? iField5 + 1 : 0
                    iField6 = iField6+1 < lenarray(gi_Arr_1) ? iField6 + 1 : 0
                    iField7 = iField7+1 < lenarray(gi_Arr_2) ? iField7 + 1 : 0
             schedule(1002, 1.0 * (60/ibpm), 0, 0, iField5, iField6, iField7)
            endif
    endin
```

Then save the loop index into a global map so that user can stop at any time.

In command:

```
removeLoop 1002;
```

In Server:

```
strInt = int(msg[0])
    if strInt in loopControlMap:
        lc = loopControlMap[strInt].lc
        cs.compileOrc(f"{lc} = 2")
```

So that the `gil1002` can be `2` to escape the loop.

4 Conclusion

While there are more things to cover in this paper, especially the 'cLoopInB' command that binds p4-field to the chord progression as intervals from the root of the chord with min and max pitch attributes. The system is also able to assign and use bus and channels dynamically among users. However, this paper needs to end here for limitation in length. For more information and source code, please, visit github link as well as demo youtube video provided in the references.

The Python-based TCP socket server implemented for live coding sessions using the Csound engine demonstrates a fun platform for real-time musical creativity and collaboration. Its client-server architecture significantly enhances interactivity, making it highly suitable for live performances and

collaborative environments. However, the reliance on a text-based interface, while simplifying operations, may also present challenges in user engagement and accessibility, particularly for those more accustomed to graphical interfaces. In future developments, enhancing the user interface with graphical elements could improve usability and appeal, making the system more accessible to a broader audience.

References

1. Lazzarini, V.: Introducing litePlay.js, https://vlazzarini.github.io/blogpost/2023/01/02/Introducing-litePlay.html
2. Yi, S.: csound-live-code, https://github.com/kunstmusik/csound-live-code/blob/main/doc/intro.md
3. Pham, H.: pyaudio, https://pypi.org/project/PyAudio/
4. SeoKyeong, K.: Source Code, https://github.com/Toby-SeoKyeong-Kim/CsoundLiveCodingwithPython
5. SeoKyeong, K.: Demo Video, https://www.youtube.com/watch?v=-5rpBh8lthw

Csound Expansion

Csound Journey in Iran

Parham Izadyar[1], Amin Khoshsabk[2] and Ghazale Moqanaki[3]

[1]parhamizadyar93@gmail.com
[2]amin.khoshsabk@gmail.com
[3]gh.moqanaki@gmail.com

Abstract. Over the past decade, the growth of Csound users in Iran has had a profound impact on the music scene, not only in the realm of electronic music but also in the general music scene. This software has empowered young composers to articulate their creative visions more effectively and more easily to perform their pieces, thereby contributing to a vibrant and evolving music scene. The accessibility of Csound, its open-source nature, and the boundless creative opportunities it offers to composers have made it a favorite companion for their music. Additionally, there is a noticeable increase in composers that utilize Csound. This innovation not only benefits composers by providing them with new tools and possibilities but also introduces fresh perspectives for listeners. It is clear that many audiences are attending more and more to live electronic music performances each year, which has enriched the connection between composers and their audience. Given these observations, it is clear that Csound has had a unique and valuable influence on the contemporary Iranian music landscape. Consequently, the aim of this article is to highlight the significance of Csound in Iranian music. To better understand the widespread appeal of Csound in Iran, some questions were written for those who have experience with Csound. Their responses in following will shed light on the positive impact Csound has had on their artistic journey.

Keywords: Csound, Iran, Contemporary music, Electronic music, Composition

1 Introduction

The contemporary Iranian composers have always been deeply concerned with the integration of Iranian classical music elements into their compositions, a task that has presented significant challenges due to the need to blend these elements with modern and contemporary techniques. But now with the help of Csound composers had found a solution, which has enabled them to express their intended emotions and thoughts more effectively through their music.

Furthermore, the convenient and easy use of Csound has made it possible to perform individual and personal performances in small spaces, leading to the spread of informal performances conducted with Csound. Moreover, Csound has emerged as a valuable tool for individuals interested in sound design for games and film music, as well as algorithmic composition.

2 When Did Electronic Music Start in Iran?

It is hard to point out a date in which electronic music entered Iran. However, in 1971, it was Iannis Xenakis who performed his electronic piece Persepolis for the very first time in "Shiraz Festival of Arts" [1]. According to Alireza Farhang: "Since the late 1960s up until the 1979 Revolution, prominent figures of avant-garde arts, among them Iannis Xenakis, Peter Brook, John Cage, Gordon Mumma, Davis Tudor, Karlheinz Stockhausen, and Merce Cunningham, participated in Shiraz Arts

Festival in Iran" [2]. Furthermore, works by various composers were featured in the Shiraz Festival, including electronic compositions by Stockhausen titled " Gesang des Jünglinge, Telemusik, Kontakte, Hymnen, etc. " [1].

But the first Iranian composer that might have been composing with electronic music is Alireza Mashayekhi who composed Shur op.15 in 1968. Later in 2005 Shur along with other pieces published in an album of electronic pieces [3]. It seems the album was an attempt to change the quality of Iranian classical instruments with electronic music. Alireza Mashayekhi is one of the notable Iranian composers that tries to find new ways to use Iranian classical music with contemporary perspectives.

Another pioneer Iranian composer who worked with electronic music is Shahrokh Khajenouri. He has been a dedicated composer of electronic music since the 1970s. Before the revolution of Iran, he used synthesizers and VCS3. Later after the entrance of computer music software he published "Dialogue for Flute and Electronic Music (1997)", which was a series of works for acoustical instruments and computer music featuring Azin Movahed as the flutist [4].

After this events, various musicians started their activities in the realm of electronic music. Among these, *Yarava Music Group* [5] stands out as a notable pioneer. Interviewing Mehdi Jalali, one of the co-founders of this Group, the Yarava was very firstly aimed at Classical Iranian music but slowly shifted to electronic music. It was in 2003 that Alireza Mashayekhi encouraged Mehdi Jalali to found the "Modern Orchestra of Yarava". Two years later in 2005 Yarava made their very first event named Music Privacy No.14 that the first signs of electronic music appeared. The event was aiming to talk about 20th century music and therefore they came across electronic music. This event started like a regular speech but turned to an artistic electronic performance [6].

3 The Situation of Electronic Music in Iran

Over the past few decades, the evolution of electronic music in Iran has been characterized by substantial contributions from a variety of composers, musicians, and ensembles. Notably, the Yarava Music Group has emerged as key figures in this development. Yarava significantly contributed to the evolution of electronic music in Iran including Csound introduction. In 2009 through an initiative, Yarava invited Shahrokh Khajenouri, Kiawasch SahebNasagh, and Joachim Heintz to conduct an educational seminar on electronic music. This event served as the beginning journey of Csound within the country, with Joachim Heintz demonstrating its capabilities through his compositions [7].

In 2015 SET festival was founded by the artist Ata Ebtekar aka SOTE. In the last two decades he became one of the very dominant characters of the electronic music scene in Iran. SET is an artist-run festival focusing on experimental music and audio-visual performances [8].

Further advancements in the dissemination of knowledge concerning electronic music production were evident in 2015, with the organization of a three-day workshop at the Tehran University. Initiated by Professor Sara Abazari and featuring Joachim Heintz, this workshop aimed to empower participants by transforming their conceptual ideas into tangible musical creations utilizing Csound.

Building upon these foundational initiatives, the Yarava Music Group, alongside Joachim Heintz, spearheaded the Seda-Tehran-Music Festival and the inaugural "Reza Korourian Award" in 2016. These events played a crucial role in encouraging emerging Iranian composers to explore electronic music and engage with international works. Subsequent collaborations between the Yarava Music Group and Joachim Heintz resulted in the establishment of the Tehran International Electronic Music Festival (TIEMF) [9] and subsequent Reza Korourian Awards, fostering a vibrant ecosystem for electronic music enthusiasts.

Throughout these endeavors, Joachim Heintz, along with collaborators such as Farhad Ilaghi Hosseini, Amin Khoshsabk, and Parham Izadyar, conducted numerous workshops—both in-person and online—focused on Csound. These educational sessions significantly enhanced the awareness and proficiency of younger generations in utilizing the Csound programming language.

4 The Effects of Csound on Iranian Musicians [10]

As mentioned before the adoption of Csound in Iran started with a series of workshops by Joachim Heintz, aimed at introducing this audio programming language to the Iranian musicians. Over time, as the number of Iranian users of Csound grew, it became increasingly popular among composers, particularly among the younger generation. However, the appeal of Csound extended beyond this Introduction. The software's open-source nature, the plentitude of online educational materials, extensive example libraries, and an active community forum, the expansion of festivals and educational workshops, coupled with its user-friendly interface and the similarity to widely known programming languages such as C and C++, made it a friendly and convenient option to learn for its users.

Furthermore, Csound proved to be an ideal language for individuals working with plugins and Digital Audio Workstations (DAWs) who were not inclined towards live performances. Users could easily develop their desired plugins using *Cabbage* [11] and incorporate them into their compositions. Additionally, those interested in pursuing electronic music as part of their academic efforts recognized Csound as a reliable option, dedicating their efforts to mastering the software.

In recent years, the use of Csound in compositional practices has significantly transformed the landscape of electronic music in Iran, offering composers a novel perspective on their works. This innovative approach is evident in their broader body of work, where Csound is not only utilized but also serves as a foundation for exploring the boundaries of electronic music. Through the algorithmic methodologies acquired through the study and application of Csound, composers have generated intriguing compositions and orchestrated a variety of performances. The use of Csound and live electronic performances among emerging composers has catalyzed a surge of activities within the electronic music domain. This evolution is discernible through of social media platforms and an increased interest among individuals in learning and engaging with this genre.

Beyond its impact on composers, Csound has also significantly influenced performers and instrumentalists, presenting them with new challenges in their performance and also their practice [12]. These challenges encompass the coordination with electronic parts, during live performances and the adaptation to electronic music notation. Moreover, musicians have perceived new understanding and acceptance of contemporary music, paving the way for continued exploration in this realm. also, the utilization of live performance technologies, such as sensors and the integration of smart phones with Wi-Fi, has facilitated audience participation, thereby creating a unique performance experience. While some innovations have been successful, others have not, the overall experimentation has empowered audiences in concert halls to engage more actively with electronic music, enhancing their familiarity with its capabilities and potentialities.

Based on the conducted interviews, the initial learning curve associated with Csound appears challenging, primarily due to factors such as the less active community relative to comparable software like *Puredata* and the lack of video tutorial resources. Nevertheless, approximately fifteen years after the initial educational workshop on Csound, the software has facilitated the creation of numerous compositions across various genres. Its versatility is demonstrated in applications ranging from creating audio files for fixed media to integrating in live electronic performances. These applications

include sound processing, algorithmic pattern utilization, and plugin implementation for instrument or voice emulation.

References

1. Gluck, R.: The Shiraz Arts festival: Western Avant-Garde Arts in 1970s Iran, p.p.: 20-28, In: Leonardo, February 2007. https://muse.jhu.edu/article/209700/pdf
2. Farhang, A.: Electronic Music in Iran: Tradition and Modernity. https://www.alirezafarhang.com/post/electronic-music-in-iran
3. According to Album Booklet, "Happy Electronic Sounds". Produced & printed in Music Center of Hoze-ye Honari, Tehran, Iran, 2005, Shur was composed at Gaudeamus electronic studio in Bilthoven and at the Utrecht Studio of Sonology in The Netherlands where Mashayekhi studied electronic and computer music and attended lectures of Gottfried Michael Koenig.
4. Dialogue for Flute and Electronic Music, https://houseno4.org/artists/shahrokh-khajenouri/
5. Yarava Music Group, https://yarava.com/
6. Interview with Mehdi Jalali (composer and founder of Yarava Music Group) by Parham Izadyar and Amin Khoshsabk February 2024
7. Joachim Heintz, https://joachimheintz.net/nav/le_yarava.html
8. SET festival, http://setfest.org/about/
9. TIEMF, https://tiemf.org/
10. This part is written according to correspondences and interviews between Parham Izadyar and composers whose names are mentioned in references
11. Cabbage, https://cabbageaudio.com
12. According to interviews done by Parham Izadyar and instrumentalists whose names are written in references.

Interviews and Correspondences

- Online interview with Sara Abazari (composer) by Amin Khoshsabk on March 2024
- Online interview with Sahar Helmi (composer) by Parham Izadyar and Ghazale Moqanaki on March 2024
- Online interview with Ali Balighi (composer) by Amin Khoshsabk and Ghazale Moqanaki on March 2024
- Interview with Aynaz Dargahi (pianist) by Parham Izadyar at Qazvin on 22 July 2023
- Interview with Parastoo Shirani (flutist) by Parham Izadyar at Karaj on 12 May 2023
- Online interview with Mehrak MalekPour (setar player) by Parham Izadyar on 16 May 2023
- Correspondences between Parham Izadyar and Vesal Javaheri (composer) from 2022 until now
- Online interview with Maryam Golrokh (composer) by Parham Izadyar on November 2021
- Interview with Erfan Javadi (composer) by Parham Izadyar at Karaj on November 2021
- Interview with Madjid Tahriri (composer) by Parham Izadyar at Art University of Tehran on September 2021
- Interview with Arshan Najafi (composer and kamancheh player) by Parham Izadyar at Tehran on December, 2020
- Correspondences between Parham Izadyar and Pendar Azimi (composer) from 2020 until now
- Several interviews with Joachim Heintz (composer) by Parham Izadyar and Amin Khoshsabk from 2019 until now
- Online interview with Ghazale Moqanaki (composer and santoor player) by Parham Izadyar on 30 November 2019
- Several interviews with Mehdi Jalali (composer and founder of Yarava Music Group) by Parham Izadyar, Amin Khoshsabk and Ghazale Moqanaki from 2017 until now

Examples of pieces by Iranian composers created by Csound

- Ali Balighi

https://soundcloud.com/alibalighi/daramad-for-3-sopranos-and-fixed-media

- Kasra Faridi:
https://stilll-off.bandcamp.com/album/counterpoint

- Parham Izadyar:
https://www.parhamizadyar.net/music/hurqelya/emulation/emulation.html
https://www.parhamizadyar.net/music/hurqelya/disappearance/disappearance.html
https://www.parhamizadyar.net/music/ymg/ymg.html
(Plugin) https://www.parhamizadyar.net/code/cabbage/cabbageVST.html

- Erfan Javadi:
https://soundcloud.com/erfanjavadi/correct-prediction

- Amin Khoshsabk:
https://soundcloud.com/amin-khoshsabk/camouflage
https://soundcloud.com/amin-khoshsabk/manifestation
https://soundcloud.com/amin-khoshsabk/words-and-mirrors

- Sepideh Yaftian:
https://beeptunes.com/track/581331953
https://beeptunes.com/track/581332008

- Soheil Zarrinpour:
https://soundcloud.com/soheilzarrinpour/resurrection-soheil-zarrinpour

- Mehrnoosh Zolfaghari:
https://www.youtube.com/watch?v=phEOn6ONrj0

For watching the video presentation of this paper by Parham Izadyar in the ICSC 2024 please go to the below link:
https://vimeo.com/1011531081

Using SOFA HRTF Files with Csound Binaural Opcodes

Thom McDonnell[1] and Dr Brian Carty[2]

[1]Sound Training College, Dublin, soundtraining.com
[2]Institute of Art, Design and Technology, Dún Laoghaire, Dublin, iadt.ie
[1]thom@soundtraining.com
[2]brian.carty@iadt.ie

Abstract. The Csound HRTF opcodes were initially written for use with a generic 'dummy head' dataset of location measurements. More recently, the field of binaural processing has enjoyed a renaissance through the proliferation of virtual loudspeaker processing. In parallel, the SOFA file format has been developed to store HRTF datasets in a defined manner. This paper discusses a method to allow the Csound HRTF opcodes to use any SOFA HRTF dataset. The outlined approach (available as a command line tool) takes any given SOFA HRTF dataset and preprocesses it to work with the existing opcodes; it essentially stores HRTFs for each location defined in the original 'dummy head' dataset used. A rigorous interpolation algorithm is used to derive HRTFs for non-measured locations where necessary.

Keywords: HRTF opcodes, SOFA, binaural.

1 Introduction: Background & Context

The field of binaural audio is predicated on how sound is altered from source to listener. This information changes based on sound location and is individual specific. The shape of the outer ear impacts incoming sound; for example, it can boost frequencies that 'fit' into the various cavities its anatomy creates (much like the various cavities in an acoustic instrument that combine to contribute to its timbre). These frequency alterations can be significant (in the context of hearing).
The terminology used for a the function that describes these alterations for a particular source location in the frequency domain is the Head-related Transfer Function (HRTF – understood broadly here as time/frequency domain information for clarity). There will be a unique HRTF for the left and right ear for any given source location relative to a listener. They are thus commonly stored in sets, containing information about various locations (i.e. the HRTF for a source at 0 degrees in front of listener for the left and right ear, then 5 degrees to the right, then 10 degrees etc.).
The individual-specific (although broadly similar) nature of the outer ear anatomy means that hearing is an individual experience; an analogy can be drawn to fingerprints. The boosts created by one ear will be slightly different in frequency and dB to another. For example, the arrow in the figure below is a slightly different length for each individual ear, leading to a different resonance.

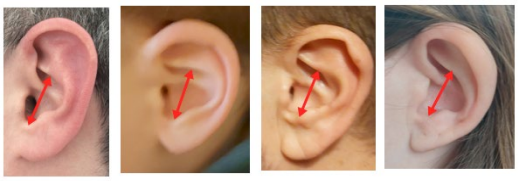


Fig. 1. The individual nature of ears.

Binaural audio is seeing a surge in interest, with commercial music mixes often now prepared in a

spatial format, as well as other applications including audio for games etc. The 'virtual loudspeaker' approach is regularly employed, whereby a multi-loudspeaker algorithm is rendered binaurally, using an appropriately spatialised virtual source for each loudspeaker location. Further detail on this introduction is available here [1].

1.1 Working with HRTFs

Capturing the HRTFs of a particular individual is non-trivial. Traditionally, the task was completed in an ideally anechoic space, with great care needed to ensure accurate and reliable results for various source locations around a listener; a tedious, expensive task not suited to the general consumer. More recently, efforts are being made to optimise the process (due perhaps to commercial applications; the personalised HRTF is an exciting prospect); this is an open research question.
Using HRTFs, a given sound source can be virtually spatialised. If the response of the ears to a sound source from a given location is known (the left and right HRTF), any sound source can be virtually spatialised to this location (an analogy can be drawn to convolution reverb here). In summary, the task is to find out how the left and right ear impact all audible frequencies for a given source location, and process the sound of interest in the same manner. It is important to play back the result on headphones, to avoid colouration by a listening environment, crosstalk and reintroducing the outer ear a second time (it has already been considered in the HRTF).
A general HRTF dataset, using average anthropomorphic measurements is perhaps a suitable starting point; however, a personalised HRTF dataset is preferable (offering a more accurate virtual spatialisation experience). Another approach may be to ask a listener to audition a number of HRTF sets and choose a best fit.
The above process works satisfactorily for a given source location (as with many such endeavours, there are many details to consider), but some manner of interpolation is needed for a moving source, assuming a HRTF dataset of a number of measured points around a listener. Further detail on working with HRTFs is available here [1].

1.2 HRTFs and Csound

Csound offers a suite of HRTF processing algorithms. These opcodes address the key issues of the field and have proven robust since release. The opcodes aim to use empirical data and offer an audio-centric approach. One of the main issues in development of a HRTF virtual sound spatialisation system, as above, is interpolation for non-measured source locations or moving sources (i.e. what happens 'in between' measurements?). This is discussed in some detail here [1]. One proposed approach involves magnitude interpolation and phase truncation; the four nearest measured (HRTF) points are considered and a suitable 'in-between' magnitude spectrum is created, which can be paired with the nearest measured phase spectrum. This approach has been shown to work well [1]. It is illustrated below. A non-measured point is derived by combining (with appropriate weighting) the spectral magnitudes of the nearest 4 measured points, and using the nearest measured phase spectrum. A moving source can continuously perform this interpolation, employing a short crossfade when the phase spectrum is updated.

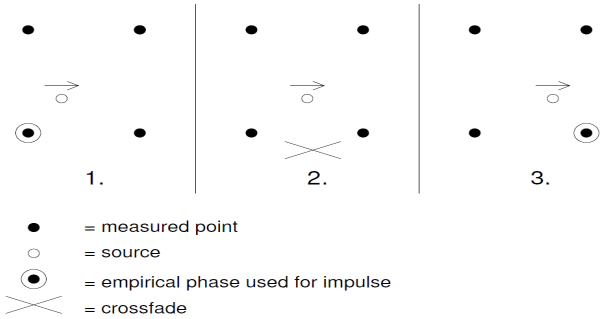


Fig. 2. Magnitude Interpolation, Phase Truncation.

The Csound HRTF opcodes were designed to use the MIT KEMAR dataset (this uses averaged measurements through a 'dummy head'). The HRTF information is passed to Csound in an optimal manner, truncating the data to only the most relevant (while keeping an empirical approach; the information is not repurposed or processed for smaller storage requirements) and storing a frequency domain representation (important to avoid time-domain interpolation issues etc.). More detail is available here [1].

1.3 SOFA and Csound

The SOFA [2] file format aims to provide a consistent way to store and distribute spatial audio information (including, but not limited to, HRTF files). This paper now discusses a proposal to allow any SOFA format HRTF dataset to be prepared in a manner that can be used with Csound HRTF opcodes.
A number of approaches to using SOFA files with Csound HRTF algorithms are possible. The below approach is proposed and presented as a command-line tool; in summary, it aims to take any SOFA HRTF set, and prepare it in a manner that works with the existing opcodes (essentially storing HRTFs for the locations defined by the originally used dataset). It is acknowledged that some HRTF datasets may offer more comprehensive/specific detail (more measured locations); it is hoped that the interpolation algorithm outlined above addresses this. The discussion section below reflects further on this.

2 Using SOFA HRTF files with Existing Csound Opcodes: Methodology

The code to process SOFA HRTF datasets for use with existing Csound opcodes is discussed in detail here [3] and available here [4]; it broadly steps through the following tasks -
Inclusion of relevant dependencies, including *netcdf* functions (SOFA is based on this file type) [2].
Key attributes of the provided SOFA file are then identified, using functions from the netcdf library. These include M (the number of source locations measured), R (the number of receivers – 2 in the case of HRTF readings for a left and right ear) and N (number of audio samples per reading). Each HRTF (/impulse response) is stored.
Spherical coordinates are insisted upon (cartesian are also possible) and a comparison of the SOFA file's stored locations with the originally used MIT dataset locations is performed. If the locations

match (i.e. the SOFA file follows the MIT dataset location system), a section of processing can now be skipped, if not, interpolation is required to align the SOFA file to the expected format.
The individual time domain impulses (representing the HRTFs) are then considered; silence is stripped from the start and the end of the audio to optimise the process. It is important not to alter the short time delays between the left and right ear responses (this represents interaural time difference; a crucial localisation cue), but silence before and after the relevant information can be removed. A simple algorithm was arrived at to implement this process; this approach was based on a review of the typical makeup of a HRTF dataset. Two non-zero samples are a key indicator of relevant information commencing.
Interpolation then occurs if required. Information for each point measured in the reference MIT HRTF dataset is prepared. A magnitude interpolation, phase truncation algorithm (outlined above) is used as required, employing the four nearest measured HRTF values.
Some inconsistencies in parsing of SOFA files were noted during development (reversals were introduced – confusions in the nearest measured points); a quadrant check was devised to ensure nearest measured points were used. As each dataset is prepared individually by whomever captures it, and many future datasets are likely to emerge, this check was deemed appropriate.
Polar spectral values for each location are stored, and the output is a left and right .dat file, ready for use with the Csound HRTF opcodes.

3 Discussion & Analysis

The code implementing the approach outlined above [4] has been successfully tested with a number of SOFA HRTF datasets.
Limitations of the approach outlined above include a lack of a comprehensively generic approach. For example, a densely sampled SOFA dataset (i.e. many more measured locations) may utilise more interpolation than is necessary (for 'in between' locations when being utilised, and indeed in storing the locations defined in the MIT dataset if these were not empirically measured), as only the MIT dataset measurements are stored. The interpolation method has proved historically robust, however. Positive results have been reported in subjective testing [1], and the algorithm has been used in Csound successfully since 2008, with a number of users getting in touch with positive feedback and projects.
Minimum audible angle and movement angle considerations are also relevant here (and are discussed here [1]); limitations of the human hearing system with respect to a moving sound source are pertinent. It is suggested that the density of the locations measured in the MIT dataset, combined with a robust interpolation method strikes an appropriate balance here.
Another point worth mentioning is that the MIT dataset assumed symmetry; personalised human datasets are likely to display some differences in the left and right ear.
An alternate approach could be to rewrite the opcodes, to allow for *any* SOFA file input. This would involve increasing Csound dependencies and potentially impacting opcode efficiency (for example, more storage would be required, a more involved initialisation process would be needed including spectral transforms and optionally many of the steps outlined above such as truncation for efficiency etc.).
The opcodes can be processor intensive when configured to consider detailed early reflections in a binaural reverb scenario for example, so although performance of a generic implementation (as above) has not been tested, perhaps the proposed set HRTF file format has merit.

The manner in which the nearest measured HRTF data is arrived at would need to be reconsidered in such an opcode update, involving a non-trivial code rewrite. It is, however, recognised that a fully general approach may be desirable for some users (i.e. load a SOFA file directly into an instance of a binaural opcode, and use it directly on its own terms).
Once data files are prepared using the code [4], users can audition these files directly in Csound, as in the code example below. In much the same way as a different audio sample can be read inline, users simply need to change the filename of the dataset they wish to use.

```
instr 1
kaz line 0, p3, 360   ;full rotation
aleft, aright hrtfmove gasrc, kaz, 0, "left_ear.dat","right_ear.dat"
outs aleft, aright
endin
```

In the case of unexpected output, suggested first steps in troubleshooting are to review the SOFA file to ensure it contains suitable data, then to consider areas of the code such as the truncation or quadrant check.

Conclusion

It is hoped that this paper and accompanying code offers an extension of the binaural capabilities of Csound through the use of any SOFA HRTF dataset. Background to the field is offered, followed by an outline and justification of the method proposed & implemented; a potential alternate approach is also discussed.

Acknowledgements

This work was supported by an IADT Masters by research scholarship. Publication of this work was motivated by Csound user Jeanette. The HRTF opcodes were initially designed to use the MIT HRTF dataset (https://sound.media.mit.edu/resources/KEMAR.html).

References

1. Carty, B.: Movements in Binaural Space: Issues in HRTF Interpolation and Reverberation, with applications to Computer Music, PhD, Maynooth University (2010), https://mural.maynoothuniversity.ie/2580/
2. SOFA: https://www.sofaconventions.org/mediawiki/index.php/SOFA_(Spatially_Oriented_Format_for_Acoustics)
3. McDonnell. T.: Development of Open Source tolls for creative and commercial exploitation of spatial audio, Master of Art (Research), IADT (2017), https://research.thea.ie/handle/20.500.12065/4038
4. McDonnell, T.: https://github.com/thommcdonnell/SofatoDAT

Bare-metal Csound

Aman Jagwani[1] and Victor Lazzarini[2] *

[1,2]Department of Music, Maynooth University
[1]amanjagwani1998@gmail.com
[2]victor.lazzarini@mu.ie

Abstract. Csound is able to target several platforms across desktop, mobile, web and embedded environments. This enables its vast audio processing capabilities to be leveraged in a wide range of sonic and musical contexts. Particularly, embedded platforms provide great portability and flexibility for users to design custom interfaces and signal processing chains for applications like installations and live performance. However, until now, embedded support for Csound was restricted to operating system-based platforms like Raspberry Pi and Bela. This paper presents our work on the development of Bare-metal Csound, extending the embedded support to ARM-based micro-controllers. We highlight the benefits and limitations of such systems and present two platforms on which we have conducted experiments - the Electrosmith Daisy and the Xilinx Zynq 7000 FPGA System-on-Chip. We also discuss potential use cases for Bare-metal Csound as well as future directions for this work.

Keywords: Embedded Systems, Bare-metal, Micro-controllers, FPGA

1 Introduction

Embedded audio programming platforms enable users to create customized tools, instruments, and interfaces for musical performance, interaction, and expression. They extend the creative process into the design of the musical instruments or tools themselves [1]. Moreover, the easy availability and affordability of programmable embedded systems allow both artists and hobbyists, as well as larger commercial and research entities, to explore this field. This underscores the significance of the programming language or environment used in these devices, particularly in the audio domain. The availability of specialized and higher-level systems like Csound within these platforms can significantly enhance audio processing outcomes and accessibility.

These platforms can be classified into two main categories, Linux or operating system-based single-board computers like Raspberry Pi [4] and the BeagleBone Black [3] and bare-metal systems or micro-controller units(MCUs) without operating systems like the STM32 [5] and ESP32 [6].

Most software that target desktop environments work seamlessly on linux-based embedded systems, allowing users to leverage existing tools like Csound, Supercollider, Puredata etc with ease. The convenience of operating systems also means that several tasks can easily leverage existing drivers, for example, for peripheral communication, MIDI or OSC. However, generally these boards are not audio specific and require external break-out boards for audio, midi etc. They also come with overheads inherent in operating systems containing functionalities that are not required for audio applications and can result in higher latency or less available computational power for real-time audio. Specialised single-board computer systems like the Bela mitigate these issues by using dedicated sister-boards and Xenomai RT Linux to optimize for real-time audio with low latency [2]. However these boards are relatively expensive.

On the other hand, inexpensive bare-metal micro-controllers have recently become very powerful and capable of performing most audio processing tasks with low latency and quick startup times [8]. The advantages of these devices include their cost, portability (due to small form factors), and specificity. Since all functionality is programmed directly onto the microcontroller, it can be optimized and tailored to perform a specific audio processing function at its highest level of capability. Furthermore, the cross-compilation process for microcontrollers can also be portable across different

* Aman Jagwani wishes to acknowledge the support of the Hume Scholarship scheme from Maynooth University.

chips with the same architecture. For example, our experience with ARM-based bare-metal MCUs showed that building Csound for one chip, such as the Electrosmith Daisy [10], enabled it to work on other chips like the Xilinx Zynq 7000 [11] with minimal adjustments. This opens up several different MCUs and boards that have ARM CPUs, such as the Teensy [12], Raspberry Pi Pico [14], and the STM32 Nucleo Boards [13], with just one build process.

However, bare-metal systems may be limited by the resources available on the MCU. Parallel or multi-core processing is also less common in these systems (except cases like FPGA-based SoCs). Another challenge with bare-metal systems is that every task needs to be implemented manually for these chips, and most programming is done at a lower level, potentially increasing development time. For example, a user may need to implement MIDI drivers individually on different microcontrollers to establish communication with their specific sets of peripherals. Similarly, functionalities that are typically managed by operating systems, such as file I/O, need to be implemented manually on microcontrollers as well. With the emergence of platforms like Arduino [15] and Daisy, as well as comprehensive vendor-supplied Hardware Abstraction Layers (HALs), a wide range of libraries and drivers are available to mitigate these issues.

Both types of platforms have their sets of advantages and disadvantages and are best suited for specific use cases. In terms of Csound and its supported platforms, it became clear to us that enabling its use on both types of platforms would be beneficial for users. Until now, Csound was only supported on Linux-based embedded systems, as demonstrated by the pioneering work done for Csound on the Bela [16] and the Raspberry Pi [17]. This limitation arose because Csound relied on the services of an operating system and libraries such as libsndfile [18], which were only supported on OS-based platforms. Now, with Csound 7, it can be built without these dependencies, enabling cross-compilation for bare-metal.

In this paper, we present our work and experimentation done to provide Csound support for bare-metal ARM [19] based platforms. We used two platforms for experimentation: the Electrosmith Daisy and the Xilinx Zynq 7000.

2 The Daisy Platform

The Electrosmith Daisy is an open-source audio programming platform based around the STM32 micro-controller [5]. It contains an ARM Cortex-M7 CPU along with several digital and analog inputs and outputs as well as other peripherals such as audio ADCs and DACs, USB serial and MIDI communication, SPI, UART Interfaces etc. It also contains different volatile and non-volatile memory locations, each of varying size and performing at different speeds, allowing flexibility in the trade offs between program size, RAM requirements and performance [20]. The Daisy also provides higher level access to the STM32 HAL through their libDaisy [22] hardware library, simplifying the programming process.

In addition to the Csound implementation, the Daisy supports programming in C++, Arduino, PureData, and Max/MSP's Gen. Each of these environments have some limitations. The C++ and Arduino programming is generally done with DaisySP [21], Electrosmith's DSP library containing DSP objects ported from different open source libraries (including Csound). The capabilities of audio programs done in this context would therefore be limited by the availability of objects in the library which may not cover all DSP or audio programming aspects. For example, there is not much support for generative techniques in this library. Surely, in a C++ environment, missing features can be implemented from scratch or ported from other libraries as done in [1], but that is not suitabe for users or artists with limited experience. Max/MSP comes with the inherent limitation of being closed source and requiring a paid license. In terms of Pure Data, the Daisy Pure Data utility pd2daisy uses the Heavy Compiler/HVCC to generate C or C++ audio code from Pure Data patches [23]. In this context, Pure Data is more like a front end, rather than an audio engine, that is used to generate the audio code. Also, all pure data objects are not supported by this compiler, so patches may be limited [24].

Therefore, Csound support can greatly enhance the Daisy platform. Firstly, Csound is a mature and complete audio programming system. It covers virtually every DSP and audio programming task that a user would require. Secondly, Bare-metal Csound builds and uses the Csound source code itself

as a static library. This enables portability of Csound programs to these platforms as well as the leveraging and consistency of almost all of Csound's features.

2.1 Development Process

The development process for Csound on the Daisy had two parts. First, we worked on simply running Csound code on the Daisy. We made a simple test program utilizing the Csound API within the Daisy C++ environment as a test to enable Csound to compile, start and perform on the Daisy. As we were conducting our experiments we had to tackle several issues to have the code running. First we had to exclude certain parts of Csound that were not suitable for bare-metal. As mentioned before this included OS-requiring dependencies like libsndfile, portaudio, portmidi, opcodes that use file input and output like ftsamplebank, diskin etc. We also bypassed other code that required OS resources, such as for instance, threading.

Next, we had to adjust our Daisy program to accommodate the Csound library's size. By default, Daisy programs are stored on the internal 128kb flash memory of the board. This is not sufficient for programs that use Csound, so we moved the program to the external 8MB QSPI flash. This provided enough storage space but comes with the tradeoff of a reduction in speed.

Lastly, the Daisy has 1MB of internal memory available to programs. This was not sufficient as well for running csound so we had to allocate our stack and heap to the external 64MB SDRAM on the board.

Once all of these steps were completed, Csound code was able to run on the daisy. We were able to confirm audio output from the Daisy while calling csoundPerformKsmps in the Daisy audio callback:

```
void AudioCallback(AudioHandle::InputBuffer      in,
AudioHandle::OutputBuffer out,
size_t                     size)
{
        MYFLT *spout = csoundGetSpout(csound);
        int    end   = csoundGetKsmps(csound);
        for(size_t i = 0; i < size; i++)
        {
                if(cnt == 0)
                {
                        csoundPerformKsmps(csound);
                }
                out[0][i] = spout[cnt] * 0.5f;
                out[1][i] = spout[cnt + 1] * 0.5f;
                cnt = (cnt + 2) % (end * 2);
        }
}
```

The next step was to create an interface between the Daisy's peripherals and Csound code so that users could work within Csound itself instead of the Daisy C++ environment with the Csound API. We used the Bela implementation of Csound as a reference for this [16]. We connected the analog inputs of the Daisy to Csound software bus channels named "Analog0, Analog1...". We also added a plug-in opcode for the Daisy digital inputs with the following signature:

```
kDigi digiInDaisy iPinNumber, iPullMode
```

Next, we connected Csound's message callback with the serial logging available on the Daisy and Csound's Midi Callbacks with Daisy's USB midi communication. It is important to note that both of these features use the USB port on the Daisy board, so only one of them is accessible at a time.

The daisyCsound interface is available at this link:
`https://github.com/amanjagwani/DaisyCsound`.

As shown in [1], the Daisy is useful for creating custom audio processing devices for live performance. The small form factor and relatively low price of this board also means that it can easily be embedded into artworks or installations and combined with various forms of interactivity through sensors. Csound support can greatly enhance these benefits of using this platform.

3 Xilinx Zynq 7000 FPGA Platform

While we were experimenting with the Daisy, it occurred to us that there are several other boards and chips that contain ARM Cortex CPUs. One of them is the Xilinx Zynq 7000, which is an FPGA-

based System on Chip(SoC). It contains two parts, the processing system(PS), consisting of a dual-core ARM Cortex-A9 CPU and the programmable logic(PL), consisting of the FPGA fabric [11]. This enables seamless interfacing between programs running on the CPU with custom accelerated hardware running on the PL.

We were able to build bare-metal Csound for the CPU on this SoC with minimal changes from the Daisy build - we just had to adjust the cross-compile instructions to target the Cortex-A9 instead of the Cortex-M7. Once this was done, we were able to test and run Csound on the PS of the Zynq, opening up several opportunities to leverage the power of FPGAs with Csound.

FPGAs allow a large amount of parallelism, they can perform ultra-low latency, sample-by-sample processing. They have a large number of GPIOs and are capable of performing extremely intensive computational tasks with high throughput [27]. These features can be extremely beneficial for complex audio processing tasks such as spectral processing, complex reverbs, huge multi-channel arrays, physical modelling etc as shown in [26] and [25].

However, FPGAs are generally difficult to program since they require specialised low-level hardware design knowlegde. One way around this issue is the use of High Level Synthesis(HLS) techniques. HLS allows FPGA hardware designs to be generated from C or C++ code. It does require some specialised knowledge and programming practices but is still more accesible than traditional FPGA programming with hardware description languages such as VHDL and Verilog [28] [27]. We use HLS to generate custom audio processing IP cores which are like modules that run on the PL. These include modules such as oscillators, envelopes, filters and reverbs.

Thus, the combination of bare-metal csound running on the PS and custom modules running on the PL presents the opportunity of using all the features of Csound with accelerated, ultra-low latency, parallel and efficient audio processing, enabled by the FPGA, in an embedded system. We demonstrate this by generating audio from Csound on the PS, passing it to the FPGA with the Xilinx DMA Controller [29] and processing it with our HLS modules. One of the modules we used for this is a port of Csound's reverbsc opcode. It runs as an IP core on the fpga, processing sample-by-sample. With the resources available on the FPGA, several instances of reverbsc can also be instantiated in parallel, enabling multi-channel processing by this complex reverb on a single chip which is similar in size to a coin, showcasing the potential power of this platform.

Similar to reverbsc, ports for different opcodes especially computationally intensive opcodes such as the pvs opcodes can be made as IP cores to offload the processing from the CPU. This can expand the audio processing possibilities of embedded systems while maintaining Csound's familiar programming environment for users.

Another way to use Csound on this platform would be to use Csound's generative possibilities such as random number generators, flexible probabilistic scheduling and sequencing to control sound synthesis running on the FPGA that leverages the high-sampling rates and parallelism for virtual analog synthesis with several voices.

4 Future Directions

One of the developments that can be done in the future is adding support for other bare-metal architectures such as the Tensilica Xtensa on the widely used and cheap ESP32 board [6]. This board specialises in wireless communication with on-board Wi-Fi and bluetooth support. Combining these features with Csound can be extremely powerful.

Additionally, there are platform specific developments that can be done as well. For the Daisy, we can use the SPI peripheral to connect an SD card to the board and then implement File I/O and include some of the opcodes that we had to exclude.

For the Zynq, we are currently manually passing audio using the Csound API from the Csound spout buffer to the DMA controller. In the future we can develop plug-in opcodes that would let users communicate with the FPGA IP cores directly from Csound code.

5 Conclusions

Bare-metal Csound's development can open up new possibilities for the Csound community in the contexts of live performance, instrument design, interactive systems and installations. Our experimen-

tation with the two platforms discussed in this paper provide good indication that having the option to run Csound's comprehensive audio processing on small and inexpensive bare-metal hardware can be conducive to several creative outputs. With Csound 7, users will be able to simply build Csound with an appropriate cross-compile file and with the bare-metal option selected and deploy their audio processing on bare-metal ARM based micro-controllers. At this stage, plans for comprehensive documentation, distribution and packaging are still being developed.

References

1. Jagwani, A.: Creative Possibilities and Customizability of Live Performance Systems with Open Source Programming Platforms. In: Proceedings of the 2nd International Symposium on Ubiquitous Music (UbiMus 2023), Edited by Azeema Yaseen, Brian Bridges, Marcello Messina, Damián Keller, pp. [133-144]. Ulster University, Derry Londonderry Campus, Northern Ireland, November 2-4, 2023. ISBN 978-65-00-85069-7. https://www.ulster.ac.uk/conference/ubimus
2. McPherson, A.: Bela: An embedded platform for low-latency feedback control of sound. The Journal of the Acoustical Society of America, vol. 141, no. 5, pp. 3618–3618 (2017)
3. BeagleBoneBlack. Available: http://beagleboard.org/black. Accessed: April 17, 2024.
4. RaspberryPi. Available: https://www.raspberrypi.org/. Accessed: April 24, 2024.
5. STM32 32-bit ARM Cortex MCUs. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html. Accessed: April 24, 2024.
6. ESP32. Available: https://www.espressif.com/en/products/socs/esp32. Accessed: April 24, 2024.
7. Lazzarini, V. et al.: Csound: A Sound and Music Computing System. Springer (2016)
8. Mulshine, M., Snyder, J.: OOPS: An Audio Synthesis Library in C for Embedded (and Other) Applications. In: Proceedings of the International Conference on New Interfaces for Musical Expression (NIME 2017), Copenhagen, Denmark, 2017. Princeton University, 310 Woolworth Center, Princeton, NJ 08544. Email: mulshine@princeton.edu, josnyder@princeton.edu.
9. Csound Github site, http://csound.github.io
10. Electrosmith Daisy. Available: https://electro-smith.com/collections/daisy. Accessed: April 28, 2024.
11. Xilinx Zynq-7000 SoC. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. Accessed: April 28, 2024.
12. Teensy USB Development Board. Available: https://www.pjrc.com/teensy/. Accessed: April 28, 2024.
13. STM32 Nucleo Boards. Available: https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html. Accessed: April 28, 2024.
14. Raspberry Pi Pico. Available: https://www.raspberrypi.com/products/raspberry-pi-pico/. Accessed: April 28, 2024.
15. Arduino. Available: https://www.arduino.cc/. Accessed: April 28, 2024.
16. Csound on Bela. Available: https://csound.com/site/news/2019/05/06/csound-on-bela. Accessed: April 28, 2024.
17. Csound 30 Years Maynooth - Workshop Csound and Raspberry Pi. Available: https://cosmoproject.github.io/Workshop_Maynooth/. Accessed: April 28, 2024.
18. libsndfile. Available: http://www.mega-nerd.com/libsndfile/. Accessed: April 28, 2024.
19. ARM Architecture. Available: https://www.arm.com/architecture/cpu. Accessed: April 28, 2024.
20. Electrosmith - Memory: What is the difference?. Available: https://electro-smith.com/pages/memory-what-is-the-difference. Accessed: April 28, 2024.
21. DaisySP - Digital Signal Processor library for Daisy audio platform. Available: https://github.com/electro-smith/DaisySP. Accessed: April 28, 2024.
22. libDaisy - Hardware library for Daisy audio platform. Available: https://github.com/electro-smith/libDaisy. Accessed: April 28, 2024.
23. pd2dsy - A tool for converting Pure Data patches to DaisySP code. Available: https://github.com/electro-smith/pd2dsy. Accessed: April 28, 2024.
24. hvcc - Heavy Compiler Collection. Available: https://wasted-audio.github.io/hvcc/. Accessed: April 28, 2024.
25. Wegener, C., Stang, S., Neupert, M.: FPGA-accelerated Real-Time Audio in Pure Data. In: Proceedings of the 19th Sound and Music Computing Conference, pp. [insert pages], Saint-Étienne, France, June 5-12, 2022.
26. Popoff, M., Michon, R., Risset, T., Cochard, P., Letz, S., Orlarey, Y., de Dinechin, F.: Audio DSP to FPGA Compilation: The Syfala Toolchain Approach. Technical Report RR-9507, Univ Lyon, INSA Lyon, Inria, CITI, Grame, Emeraude, May 2023. Available: https://inria.hal.science/hal-04099135.

27. Jagwani, A.: Developing a Modular Sound Synthesis Platform for FPGAs with High Level Synthesis Techniques. Master's thesis, Maynooth University, Department of Music, 2023. Supervisor: Prof. Victor Lazzarini.
28. Kastner, R., Matai, J., Neuendorffer, S.: Parallel Programming for FPGAs. ArXiv e-prints, May 2018. Available: `https://arxiv.org/abs/1805.03648`.
29. Xilinx AXI DMA IP. Available: `https://www.xilinx.com/products/intellectual-property/axi_dma.html`. Accessed: April 28, 2024.

Integrated Csound 1

Exploring the Expressive VR performance of Csound Instruments in Unity

Ken Kobayashi

Berklee College of Music
kkobayashi4@berklee.edu

Abstract. Electronic music instruments have revolutionized musical performances. These gadgets allow musicians to perform using sound synthesis, unlocking infinite possiblities from countless algorithms, from those explored thoroughly to the cutting edge. However, as sound synthesis technologies evolve, such digital instruments must also be reimagined. A instrument that fully utilizes the capabilities of modern synthesizer technology should allow one to perform not just novel sounds, but be more expressive with their performance. This paper explores such expressiveness through an instrument created in VR, the *Laser Synth*.

Keywords: CsoundUnity, VR, Performance, Csound, Unity

1 Introduction

The *Laser Synth* [1] is a VR instrument focused on playability. It is distinct from VR installations, which enable a user to create evolving soundscapes with high degrees of randomness, such as those based on physics or hordes of objects. Instead, the *Laser Synth* will be playable in the traditional sense; a performer plays with intent for specific rhythm, pitch and synth modulations.

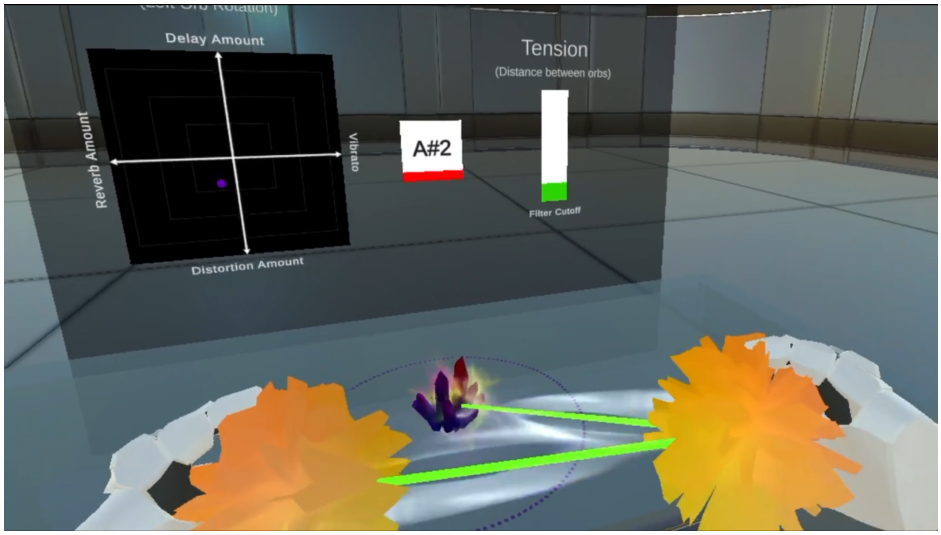


Fig. 1. *Laser Synth* being performed. Shown are both orbs in hands and the UI.

2 Instrument Control

The instrument consists of a stationary crystal and 2 virtual orbs, one to be grabbed by each hand. (See Fig. 1) The relative position of these 3 objects, along with the rotation of one, controls all elements of the instrument. The positional relationship of the 3 objects are shown by the laser connecting them. A laser connects the crystal and right orb, and the 2 orbs.

2.1 Pitch

Accurate pitch control is crucial for a melodic performance. Streamlining it acts as a neccessary foundation for all other features.

The distance between the crystal and the right orb controls pitch. The crystal remains stationary, while the right orb's position determines pitch, similar to a theremin. Simplifying this control by narrowing down the scope to a single element eliminates randomness, allowing performers to focus solely on their right hand's position for pitch adjustment.

To futher assist the performer for an accurate performance, the pitch is quantized to the 12 tone equal temperament scale. Because of the positional control, it is near impossible to play pithces accurately in a 3D, virtual space. Pitch quantization gives the performer significant margin of error to play their desired note. To compliment this, a visual tuner is shown in the center of the UI in front of them. The pitch is shown, with filling bar that represents how close to the next note it is. As shown in Fig. 1, the tuner shows A#2, with the bar being low and red, showing the performer is close to going down a pitch to A2 if the right orb is brought closer to the crystal.

Finally, to make transitions between pitches smoother, a portamento was added. This is to address the issue with large jumps in pitch being difficult to make sound elegant. It wasn't physically feasible to move fast enough. The portamento smooths out the frequencies between pitches, so the performer doesn't need to move too quickly, and can prioritize pitch accuracy.

2.2 Filter

Filter control is an invaluable tool to have in a live performance. The distance between the two orbs controls the filter cutoff frequency. This use of relative positioning of the two orbs makes filter control intuitive for the performer. If they intend to keep the filter constant through pitch changes, the performer will move both arms in identical motion. To open the filter, they will open their arms. Leaving the left orb stationary will provide a natural key mapping, opening the filter as the pitch increases.

Similar to pitch, the filter control has visual aids for the performer. On the right side of the UI (See Fig. 1), a bar reflects the cutoff of the filter. In Fig. 1, the green bar on the right is very low, reflecting the close distance between the two orbs. In addition, the color of the laser between the orbs will turn red as the filter nears the maximum frequency cutoff.

2.3 XY-Pad

For a truly expressive instrument, only being able to control a filter is not satisfactory. The rotation of the left orb controls an XY pad, enabling up to 4 parameters to be adjusted in real time. The axis (X and Y, both positive and negative) can indiviually be mapped to any parameter (See Fig. 2). The *Laser Synth* currently features 10 different parameters, including vibrato, pitchbend, reverb, and distortion (See Fig. 3).

The XY-Pad also has a visual aid for the performer. Located on the left side of the interface, a XY coordinate plane is labled with synth parameters along each axis (See Fig. 1). In Fig. 1, a dot can be observed to the bottom left of the origin of the pad. This is the value outputted by the rotation of the left orb pictured.

Combined with the filter control, the XY-pad feature adds immense expressive ability through the left hand. To manipulate the sound, the performer can use smooth and fluid gestures, such as a shoveling motion, or twisting your wrist in a figure eight motion. Simple gestures as such affect the filter and XY-pad simultaneously, resulting in an expressive sound, while being responsive to the movements of the performer. The *Laser Synth* is an virtual Csound instrument that is emotive with intension of the musician.

3 Implementation

The instrument was built using mainly two programs: Unity game engine and Csound. Some accompanying technologies were used to supplement and hasten the development process.

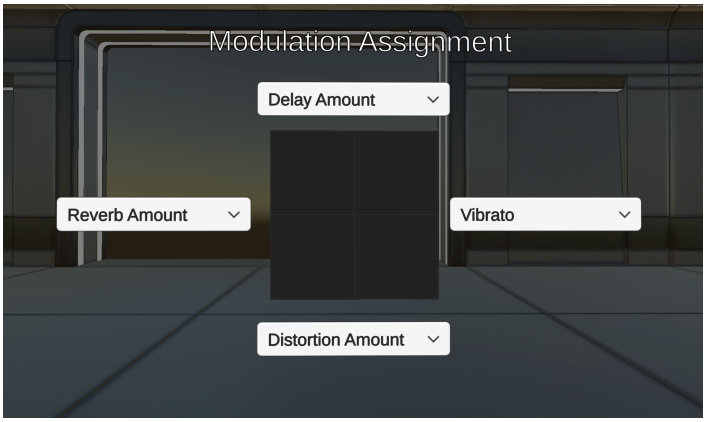


Fig. 2. XY-Pad customizable in VR through dropdowns on each axis.

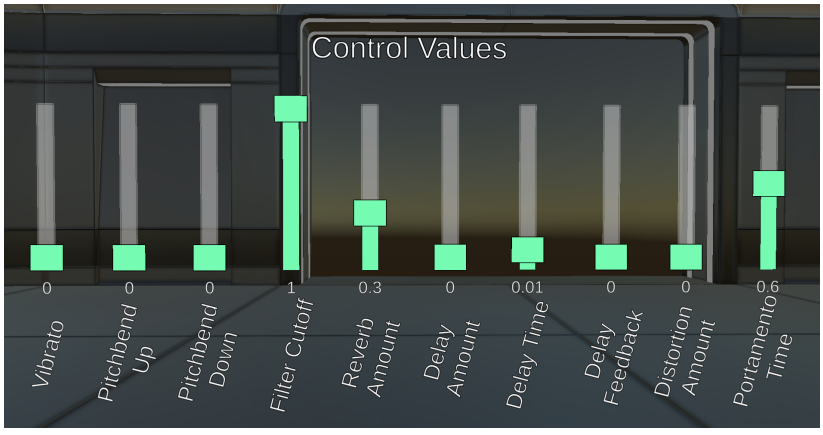


Fig. 3. List of all synth parameters and their respective values, shown in VR.

3.1 Unity

Unity is a game engine, commonly used for game development purposes. For the *Laser Synth*, Unity was used to create the 3D virtual environment. The visual design of the program, as well as the logic of the controls were created inside of Unity.

3.2 Csound

Csound drives nearly all of the sounds produced by the instrument. The program runs a monophonic lead sound, using the *poscil* opcode.

```
aOut poscil kamp, kfreq * kpitchbend + ksine, 1
```

The Cabbage editor [3], created by Rory Walsh and others, was used, allowing easy testing and implementation. One element heavily tested through Cabbage was the *note on* for the synth. A Csound channel controls turning on and off the synth. This was achieved through a listener instrument triggering a Csound event. By using Csound events, the synth is able to use the *madsr* opcode to implement an evelope, and allows easy implementation into Unity.

```
// When the noteon channel value is changed to 1, create an instance of intrument 1.
// When changed to 0, the instance is ended.
 instr 22
   knoteon = chnget:k("noteon")
   ktrigger = changed2:k(knoteon)

   if ktrigger == 1 then
       if knoteon == 1 then
           event "i", 1, 0 , -1
       else
           event "i", -1, 0, 1
       endif
       ktrigger = 0
   endif
 endin
```

All audio effects and their parameters were exposed to live manipulation by Unity through Csound channels.

```
kdist = port:k(chnget:k("distortion"), 0.01)
```

In above example, Unity is able to change the distortion amount of the Csound instument through the *"distortion"* channel. The *port* opcode is often combine with the *chnget* opcode to smooth out input values.

3.3 CsoundUnity

To allow the Csound instrument to play live audio and interact with Unity, CsoundUnity, created by Rory Walsh and others, was used. Through CsoundUnity [4], the Csound instrument is packaged within the Unity project, performing real-time audio synthesis. More importantly, it implements the *SetChannel* method into Unity to send values to the Csound instrument.
`BeamManager.cs (Ln 137)`

Setting frequency of Csound instrument through Unity

```
// convert midi note to frequency and send to csound
if (noteOn)
{
 freq = MidiToFrequency(midiNote);
 csound.SetChannel("freq", freq);
}
```

When a note is played, the played MIDI note value is converted into Hz. This value is then sent to the "freq" Csound channel to be used as the pitch.

3.4 Other Tools

Many other programs and assets were used to expedite the development process. Many of the 3D objects, visual elements, and packages are purchased from the Unity Asset Store. One such package is Auto Hand VR [5]. It handled the implementation of the VR headset and controller inputs, as well as interacting with objects through VR. It is an invaluable reasource, along with all of the other Unity Asset Store items which saved much time and effort for the project.

4 Conclusion

Expressiveness of an instrument is its ability to fulfill the performer's intention. The controls must not impede on the performer and be natural to use. In addition, it must provide a form of expression through easy, real-time parameters. The *Laser Synth* was designed with both rules in mind, making it a vessel for musical expression. The instrument is planned to allow for the sound generator to be customized for drastically personalized sounds, as well as tools for new users to understand the instrument, such as presets to jumpstart them into the experience and tutorials on the possibilities of the *Laser Synth*. This iteration of the *Laser Synth* is the start of a study of *how to play sound*, and how virtual instruments can be designed for different performances, performers, and sounds.

References

1. Ken Kobayashi. VR Laser Synth Github Page. `https://github.com/kencula/VRLaserSynth`
2. Boulanger, R.: The Csound Book: Perspectives in software synthesis, sound design, signal Processing, and Programming. MIT. (2000)
3. Cabbage Website, `https://cabbageaudio.com`
4. CsoundUnity Github site, `https://github.com/rorywalsh/CsoundUnity`
5. Auto Hand VR Unity Asset Store page, `https://assetstore.unity.com/packages/tools/game-toolkits/auto-hand-vr-interaction-165323`
6. The Canonical Csound Reference Manual, `https://csound.com/docs/manual/index.html`
7. The Csound FLOSS Manual, `https://flossmanual.csound.com`

Exploring Interactive Composition Techniques with CsoundUnity and Unity

Xiaomeng (Susan) Zhong

Berklee College of Music
UC Santa Barbara
zhong0xm@gmail.com

Abstract. This paper presents different techniques and systems that were used to create an interactive composition using Csound, Unity and CsoundUnity. The paper discuss the creation of compositional and performative systems designed by combining the synthesis powers of Csound and the interactive game mechanisms in Unity. These systems includes: generative music with logic in C# played using Csound Instruments, trigger based control systems mimicking MIDI note on/off events using Unity's collision and rigidbody mechanics, transform object and controllers functioning as real-time controls like knobs and sliders. Taking advantage of both systems, it became possible to create a game-like composition *la forêt*.

Keywords: Csound, Cabbage, Unity, CsoundUnity, C#

1 Context

la forêt is an interactive music composition created for those needed a calm musical place to escape. It was made using Csound, Unity and CsoundUnity. Csound is the worlds most extensive audio programming language with unrivaled audio quality. Unity is a game engine that comes with a strong scripting API that contains complex mechanics that can be used simply, it also allows writing of custom C# scripts that determine game behaviors and mechanics. CsoundUnity is a wrapper that was written by Rory Walsh, and is based on Richard Henniger's Csound6Net for Unity, which is a cross-platform game engine. All these elements provide flexibility and endless possibilities, and allowed me to create *la forêt*, a CsoundUnity-based interactive composition that features generative harmony and melody, trigger based and continuous musical control system, as well as audio reactive environment.

2 Generative Music System

In Csound, generative music is created using opcodes that generates new score events following a logic that determines the harmony, melody, and rhythm. Where rhythms are defined using metro, harmony and melody are defined using an array of possible notes. Following the same logic, the generative music system in *la forêt* was created in Unity using C# code, which allowed actions like key change, identifying key center and stating melodic sequence straightforward. A generative harmony system was created using the Markov chain, which is a stochastic model that decides a sequence of possible events based on probability of each. This was done by creating a 2D float array matrix that indicates the probability of the next chord progression for each tonality. This matrix wass then developed with additional arrays and functions to create the generative harmonic system. As the harmony is determined, the key center for melody wass chosen and passed on to the melody script, where, depending on the type of chord harmonically, MIDI values are added to the root note, to ensure the melody will always harmonize with the chord. The chosen harmony and melodic sequence were then stored as string variables. Different mlodic sequences were built using logic and delay was used to achieve the goal easily. The tempo is controlled using a coroutine, which pause and resume these actions at specified time intervals. The action of picking chords, melody and anything additional will be done in this coroutine. All the logic was completed using C#. Csound then received a simple score event that uses the function: *SendScoreEvent(string scoreEvent)* in the format of *"i\ "instrName\ "startTime duration p4 p5 ..."*.

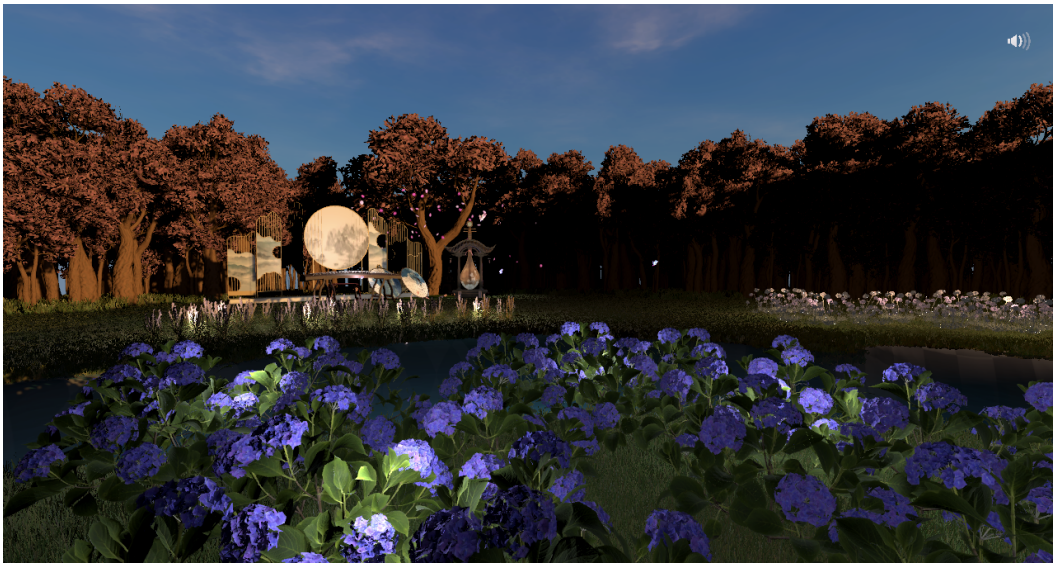


Fig. 1. la forêt

```
string[] chords = { "imaj", "iimin", "ivmaj", "vmaj", "vimin" };

float[,] harmonyMatrix = new float[,]
{
    {0.0f, 0.15f, 0.4f, 0.3f, 0.15f},
    {0.2f, 0.0f, 0.3f, 0.5f, 0.0f},
    {0.4f, 0.0f, 0.0f, 0.4f, 0.2f},
    {0.6f, 0.0f, 0.1f, 0.0f, 0.3f},
    {0.1f, 0.4f, 0.3f, 0.2f, 0.0f}
};
```

Fig. 2. Markov chain harmony matrix

3 Trigger-based System

In *la forêt*, most interactions with the environment were created based on the trigger systems. Unity has a built-in trigger and collision detection system that functions when a collider component and a rigidbody component is attached to the game object. Any objects that carry these two components can interact with other game objects that also carry those, or with controllers, giving a similar idea as pressing a key on a keyboard. The trigger message was then used to either send a score event or updating a k-rate variable through CsoundUnity's *SetChannel(string channel, MYFLT val)* function. Just like in Cabbage, a k-rate variable is linked to a string channel via *chnget*, the string channel in Csound/Cabbage must share the exact same name as the one in Unity, the MYFLT value will then be passed through to the matching k-rate variable. In *la forêt*, each type of flower represents a different instrument and a different melodic sequence. As the player approaches the flower bed, a new melodic sequence will be triggered. When the player interact with the flower, events like playing an additional note, changing the melodic sequence, or changing the timbre will occur. The scene also features two traditional Chinese instruments, the Pipa and the Guzheng. Pressing on the string will trigger a note, and depending on the relative position being pressed, the pitch of the note varies.

```
private void OnTriggerEnter(Collider other)
{

    if (csoundUnity != null)
    {
        // Pick a random note from the melodyNotes array
        int randomNote = melodyNotes[Random.Range(0, melodyNotes.Length)];

        // Play the selected random note
        PlayNote(randomNote);
    }
    else
    {
        Debug.LogWarning("CsoundUnity reference not set in Note_1.");
    }
}

void PlayNote(int note)
{
    // Play the note using CsoundUnity
    string scoreEvent = string.Format("i\"SYNTH\" 0 2 {0}", note);
    csoundUnity.SendScoreEvent(scoreEvent);
    Debug.Log("Playing note: " + note);
}
```

Fig. 3. Sending Csound score event using trigger system

4 Designing Real-time Control Tools

Real time control systems function in the same way as knobs and sliders. In Unity, there are many mechanics that can be designed for real-time controls, such as the relative position, rotation, and scale of a gameobject, and relative cursor position on screen. The position change and rotation angle of an object can be identified using the Transform component, a component that exists on all game objects. The mouse position can be coordinated using Unity's *Input.mousePosition*. Once the minimum and maximum of the game object is identified, it can then be scaled to an appropriate range that matches with the target k-rate variable range in Csound. These functions would be put within Unity's *void update()* ,which are being called every frame and thus can vary depending on device's capabilities. New values are sent using *SetChannel(string channel, MYFLT val)* controlling musical parameters in real time, like reverb size, filter cutoffs, or amplitude and frequency, allowing a whole new level of musical interactivities.

```
//Receiving rotation informations
float pitch = transform.rotation.z; //0 to -0.3, pitch
float roll = transform.rotation.x;  //-0.3 to 0.3, amplitude
float yaw = transform.rotation.y;   //-0.7 to 0.7, filter cutoff

//Scaling the value using a different method
targetFreq = MapValue(pitch, 0f, -0.3f, 300f, 200f);
currentFreq = Mathf.Lerp(currentFreq, targetFreq, lerpSpeed * Time.deltaTime);
pan = Mathf.Min(Mathf.Max(MapValue(roll, 0.3f, -0.3f, 0f, 1f), 0f), 1f);
lpcutoff = (currentFreq + Mathf.Min(Mathf.Max(MapValue(yaw, -0.7f, 0.7f, 400f, 2000f), 400f), 2000f));

//Feeding into csound
csoundUnity.SetChannel("s_freq", currentFreq);
csoundUnity.SetChannel("s_pan", pan);
csoundUnity.SetChannel("s_lpcutoff", lpcutoff);
```

Fig. 4. Real-time continuous control using transform object.

```
// Update is called once per frame
void Update()
{
    TrackMouseInput();
    RealTimeCTRL();
}

void TrackMouseInput()
{
    //Get's mouse's relative position
    mouseX = Input.mousePosition.x / Screen.width;
    mouseY = Input.mousePosition.y / Screen.height;
}

void RealTimeCTRL()
{
    //Scale it to appropriate range, sends to csound
    float cutoff = Mathf.Lerp(10000, 100, Mathf.Clamp01(mouseX));
    float pan = Mathf.Lerp(0, 1, Mathf.Clamp01(mouseY));

    csoundUnity.SetChannel("cutoff", cutoff);
    csoundUnity.SetChannel("pan", pan);
}
```

Fig. 5. Real-time continuous control using mouse position

5 Controlling Game Parameter Using Audio

Part of creating an immersive environment involves creating an interactive audio system. In the piece, Csound was also used to create procedural generated ambiances, including wind and crickets. The procedural wind was used to control both audio and visual feedback in the scene. The procedural wind instrument was made in Csound, then the control parameters was done in C#. These control parameters includes amplitude, filter cutoff frequency and filter resonance. The values were then passed on to control game mechanics. The amplitude and cutoff frequency was used to control the speed and strength of the wind, that had an affect on the grass and leaves in the scene. The amplitude was also used to control the volume of the rustling leaves samples.

6 Beyond Sound

All these control systems create endless possibilities on the types of interactive compositions, allowing re-creation of real-life instruments in a digital space, or designing new ways to compose. With Unity being a game design platform, artists can create more than just a sonic environment. Visual elements can also be designed, adapting to real-time audio changes and a different interactivities can be created based on Unity's game mechanic systems, allowing artists to create a more harmonious interactive audiovisual experience. These techniques not only allowed the creation of la forˆet, but have also been employed in a research project at MIT Nano Lab. Currently, they are being used to facilitate the recreation of the 'classic' Csound piece - wiiSoundQuest in VR that will feature a new form of interactive visual score whose gestures, contours, moving lines and shapes the players will trace, imitate, and follow.

References

1. Boulanger, Richard. The Csound Book - Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming
2. Ferrone, Harrison. Learning C# by Developing Games with Unity 2021. 6th ed.
3. Cabbage Audio, `cabbageaudio.com/`
4. The Canonical Csound Reference Manual, `csound.com/docs/manual/index.html.`
5. CsoundUnity, `rorywalsh.github.io/CsoundUnity/#/`
6. Unity - Scripting API, `docs.unity3d.com/ScriptReference/`
7. Garden Instruments and Scripts, `https://github.com/gwichanist0v0/Garden_ScriptInstrument.git`

Csound in the MetaVerse – From Cabbage to CsoundUnity and Beyond: Developing a Working Environment for SoundScapes, SoundCollages, and Collaborative SoundPlay

Hung Vo (Strong Bear)[1] and Richard Boulanger[2]

[1,2]Berklee College of Music
[1]sbear@berklee.edu
[2]rboulanger@berklee.edu

Abstract. *Csound in the MetaVerse* is an immersive multiplayer system built in Unity for Meta Quest XR headsets that supports new ways to interact with Csound instruments and effects. Players are collocated into shared physical or virtual spaces, either locally, playing together in the same physical space, or remotely, joining in with other players over the internet. In these VR and AR worlds, sounds appear as physical objects that players can hit, grab, stretch, squeeze or toss away while they continue sounding and wandering freely on their own. One can also 'connect' to the sounds via 'cords' and control individual or multiple parameters with buttons or physical gestures. This systems offers new ways to play with sound in time, to play with sounds in space, and to play with each other's sounds. And in this paper, we will highlight small excerpts from the code that provides the means for some of the more exciting, unique and important features that enhance the capabilities of CsoundUnity and make possible some of the uniquely powerful modes of interaction and collaboration that our *Csound in the MetaVerse* environment offers.

Keywords: Unity, CsoundUnity, Meta Quest, VR, AR, XR, Cabbage, ZeroTier, Immersive, Multiplayer, Collocation

1 Introduction

How to play a sound, how to shape a sound, how to control a sound, how to organize them, arrange them, connect, compare and contrast them, how to move, choreograph, and spatialize them, how to arrange them into comprehensible and meaningful units, structures, and architectures that move in time, that evolve in time, that develop over time, that speak to us, that inspire us, that resonate with us and through their divine design, speak to others in the same profound and deep way as they do to us? Csound has remarkable depth, richness, pedigree, legacy, and history. And the community has shared many brilliant inspiring and unique answers to many of these questions, and yet, the dream remains. Via *CsoundUnity*[3], integrating Csound into fully immersive, haptic, 3D VR/XR environments, such as *Unity*[4], one can now literally touch, mold, move, morph, mutate, shake, locate, and animate sound objects with virtual hands and populate imagined worlds with them. *CsoundMeta* is our first step into this new world of collaborative SoundScaping with Csound. The system continues to evolve and we are now starting to make music in there, and enjoy jamming in there, and are starting to compose in there, and share some of these ensemble performances in concert. This paper is meant to give the reader an overview of the features we have designed in the system to make it possible for one to go beyond playing in there and to begin composing in there. We will be sharing our latest *CsoundMeta* APKs, and code so that you can explore this new world along with us and share your suggestions and your creations. We will also be offering a complete set of video tutorials as well.[1] It is our hope that you will find it as exciting as we do and that you enjoy SoundSculpting as much as we are.[1]

[1] The player/composer/sound-designer immerses themselves in our SoundWorlds and plays our SoundObjects in our AI generated SoundSpaces via *Meta Quest* XR headsets[5].

2 Getting Started: Host or Client?

When you launch *CsoundMeta* you decide if you want to be the Host or a Client, and then you choose from one of our stored scenes – playing in either "Passthrough mode" (real world) or in "World mode" (*Blockade Labs* AI generated 3D Skyboxes)[8]. Then you use the mallets in each of your hands to strike, grab, stretch, squeeze, turn, toss, retrieve, create, share or delete solid (*Orb*) or open (*Wanderer*) spheres as shown in the four panes of Figure 1.

Fig. 1. Chose role (host), Choose Scene (2+Pass), Play Orbs/Wanderers (in the hall or on the street).

3 Choosing a World, an Instrument, and Saving a Preset or a Palette

Once you are in the system, you can pull up a menu to select new instruments for your orbs to play, choose a new world in which to be immersed, load or save presets or scenes, and/or add presets to a palette (a preset queue associated with each orb/wanderer that look like little pills that orbit the orb and support sequential or random recall). These menus are shown in the four panes of Figure 2.

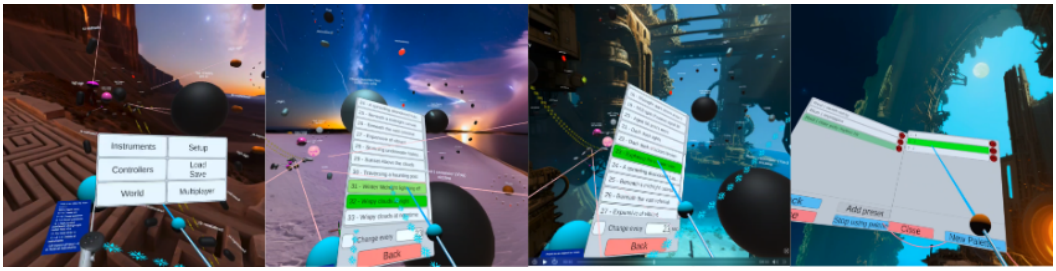


Fig. 2. Main Menu. Choosing two different worlds. Adding presets to a Palette.

4 Building a Unity Sound Settings UI from a Cabbage File

The combination of *Cabbage*[2] and *CsoundUnity* make it simple to implement a UI interface for sound settings in *Unity*. As shown in Figure 3, you can see the selection of an instrument and all the setting that can be assigned, customized, and saved, in the system.

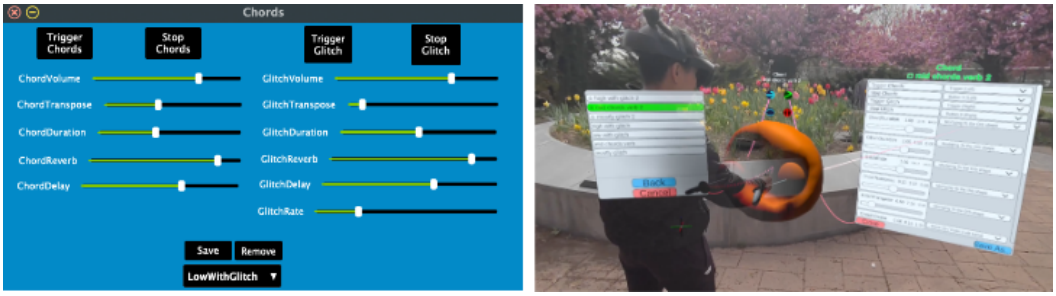


Fig. 3. Cabbage Instrument with Triggers, Controllers, Presets imported in *CsoundMeta*.

5 Supporting Multiple Players in the Same Space and Sharing Sounds

To support multiple players *Mirror Networking*[7] was used. Advantages are that no dedicated server is required, and it works best over a LAN, and even though it is server-client architecture, one of the clients can also be a server. Furthermore, *Mirror* is open source and free. [2] The *CsoundMeta* system has been tested with 11 students in classroom and labs at Berklee, and with several local users in the studio of *BoulangerLabs* joined by remote users from Hong Kong, Ohio, and LA, and in the park with family and friends. Note that when collocating with players remotely, the *CsoundMeta* system runs over a *ZeroTier* VPN [6].

Fig. 4. Multiplayer in the Berklee Labs

6 Controlling Unlimited Parameters with only One Object

`Orbs` and `Wanderers` are two special objects in our *CsoundMeta* system. The shape of an `Orb` or `Wanderer` can dictate the timbre of an assigned sound. Each *Csound* parameter is mapped to a point on the surface of the object, and the distance from the mapping point to the center of the object is correlated to the parameter value.

Fig. 5. Orbs and Wanderers being resized and deformed, i.e. sonically mutated.

7 Mapping Controller Movements to Csound Parameters

For real-time control, all Quest buttons and controls are mapped to Csound. Also, there are six different movements on each hand that are mapped as well. They are calculated relatively to the head position, which is the main camera, and the interacting object position. The six 'whole arm' movements are: up/down, forward/backward, left/right; and the six wrist movements are: up/down, left/right, and twisting. Once assigned, connecting patch cords to the Orb activates the settings as shown in Figure 6.

[2] Options such as *Photon*, *Normcore* need a dedicated server, and depend on the Internet, which, because of latency, is not the best for music jamming.

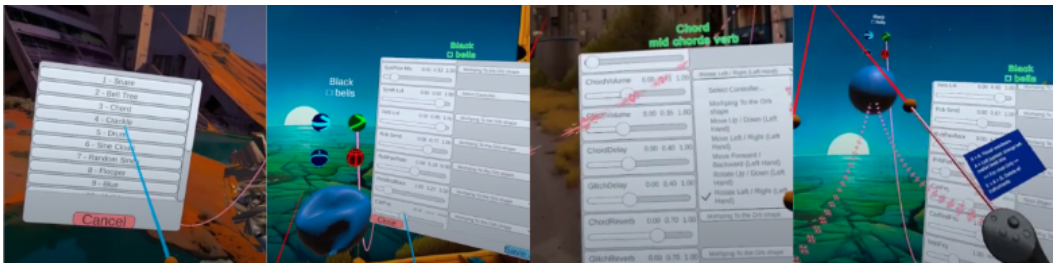


Fig. 6. Select preset. Edit preset. Map gestures. Connect *PatchCords* to activate settings.

8 Associating Multiple Objects with a Single Orb - Palettes

Designing a wide range of sounds and jamming with them is a lot of fun, but it can all get pretty chaotic when you are playing with 6-12 students! And so, to give the soundscape a more structured unfolding, and to better control the counterpoint of textures and timbres that were sounding simultaneously, a system was devised to support the attachment of a series of sound objects to a single orb/wanderer. These 'satelite-banks' of presets, which can be seen in the 4 panes of Figure 7 and appear as little pills orbiting around the orb/wanderer, can be selected individually, sequentially, or randomly. Note also that the player can toggle the shortcut/control/instruction menus, in blue, on/off at any time.

Fig. 7. Creating Palettes - Satelite-banks of presets associated with each orb.

9 Next Steps

We are designing and adding more instruments. We are in the process of designing a collection of processing objects that can wrap around an orb, (like the rings of Saturn), and add controllable post-processing. We are adding more 'traditional' ways to 'play' the Csounds – pads, buttons, grids, slider-banks, keyboards, etc. We are enhancing the system to import *.csd* files and have them magically become an orb. We are enhancing the system to allow users to import samples, loops, and audio files to use as backing tracks and ultimately post-process with our 'ring' effects. Our orbs and wanderers already respond to the audio and listen to and flicker to the spectrum of the sounds, but we are working with *Blockade Labs*[8], (whose *Skybox AI* was used to generate our current worlds), 1, to have the worlds respond to, regenerate, evolve, age, and mutate, based on the sounds; 2, to have the look and character of the orbs generate and mutate according to the quality and character of the sounds; and 3, to have sounds populate the worlds that might be associated with the visual characteristics created by the AI - creating a situation where the AI environment and object generator is listening, reflecting, reacting, and responding - is more alive to what is happening sonically.

References

1. CsoundMeta Code and Tutorials in Unity, https://www.dropbox.com/scl/fo/dndzshyljhqri55un43ci/ACC3QEnyfEQlaIi7WyjQqI0?rlkey=1zjs3hc3wuueqbgi7ca6bjvg2&dl=0
2. Cabbage, https://cabbageaudio.com/docs/introduction/
3. CsoundUnity, https://github.com/rorywalsh/CsoundUnity
4. Unity, https://unity.com/
5. Meta Quest, https://www.meta.com/
6. ZeroTier, https://www.zerotier.com/
7. Mirror Networking, https://mirror-networking.gitbook.io/docs
8. Blockade Labs, https://www.blockadelabs.com/

Face Tracking with CsoundUnity: Converting Smiles into Sounds

Bethanie Liu

Berklee College of Music
bliu3@berklee.edu

Abstract. Csound has been widely used for sound synthesis and live performance. While much exploration has been done in expanding the potential of music-making with Csound, few studies have looked into developing Csound-based music-making tools for people with physical conditions and/or disabilities. This paper presents a preliminary design and implementation of a face tracking-based musical expression system utilizing CsoundUnity's sound design capabilities for real-time musical performance. The goal of this development is aimed towards providing alternative methods for people with limb motor impairment to express music through facial gestures. Users could control parameters of Csound instruments through facial movements such as but not limited to opening their mouths and winking. The paper will also discuss observations from user testing sessions with patients at a rehabilitation facility.

Keywords: Csound, CsoundUnity, Face Tracking, Gesture-based Interaction, Disabilities, AR, Real-Time Interactive Music System, Unity

1 Context: Enabling Music-Making for People with Physical Disabilities

As a performer and researcher, the author aspires to develop real-time performance technologies that enable people with disabilities to play music and experience the joy of music-making. Individuals with hand motor impairment may face constraints in performing musical instruments that require two hands, such as the flute and violin. While there are alternative models of these instruments designed for single-handed players[1], the author aims to develop a versatile system through music technology, such that 1) Users are not limited to performing single-line melodies or chords; they can also perform textural layers of ambient soundscapes. 2) Users could simultaneously perform multiple timbres, with different facial parts each triggering different Csound[1] instruments. 3) Users could trigger the start and stop of audio samples. 4) Users could control audio effects such as but not limited to filter, reverb, delay through changing the extent of facial movement (eg the level of mouth opening).

There is a vast array of motion tracking sensors developed and applied in the field of arts, from choreography to musical expression. These systems track body movements in real-time, enabling music to be performed through hand gestures[2] and dance moves. There are also notable works researching the use of facial actions for musical expression, such as the Sonifier of Facial Actions (SoFA) system by Funk, Kuwabara and Lyons[8]. Inspired by related works, the author designed a system that uitilizes CsoundUnity's [4] sound design capabilities and Unity's interactive mechanisms to offer alternative performance methods for people with limb motor impairments.

2 System Requirements and Specifications

The preliminary design involves Unity's Live Capture package[6], ARKit XR Plugin[7], and Face Capture[3] for face tracking. The sound design components of the system are powered by CsoundUnity[4], an audio middleware for the Unity game engine. To run the system, the user would need two devices (as shown in Figure 1), one with a front camera and Face Capture; the other running the Unity project. The first device captures the user's facial expression. Once connected to the same WiFi, the

[1] An example of alternative model is the one-handed flute made by Maarten Visser of Flute Lab.
[2] An example of systems that empower music creation through movement is the MiMu Gloves created by Imogen Heap and a group of developers.
[3] Face Capture is an application by Unity and is available on app store in mobile devices

device running the Unity project will reflect the changes in the user's facial expressions. Game objects are attached to various facial parts, and scripts have been attached to game objects to enable audio control through Csound Unity.

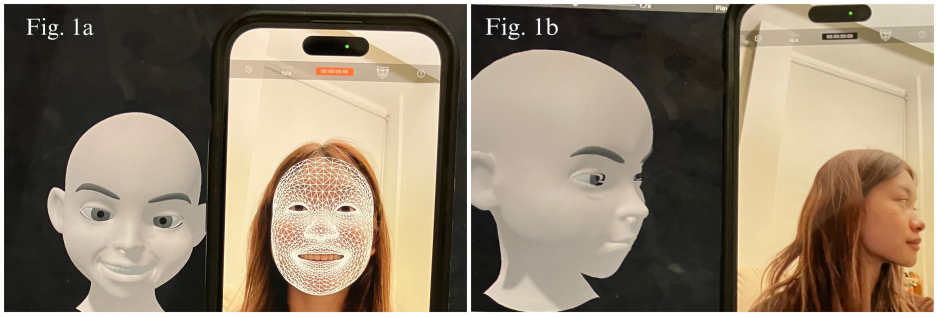


Fig. 1. A demonstration of the system specification. Using two devices, the first one being a phone with camera running Face Capture, the second one is the laptop running the Unity project. The system could be used with or without wireframes on the user's face. Fig. 1a displays the real-time reflection of the user's front profile on the avatar. Fig. 1b shows the avatar mirroring the user's head turning to the side.

Current recognisable facial movements include sideways head movement, upwards and downwards head movements, eye wink, yawning, grinning, and eyebrow raise. These movements have been categorised into continuous and discrete movements. Continuous movements are facial movements that could be done slowly, allowing for a range of changes in the XY positions of game objects to be tracked. An example of continuous movement is yawning, in which the mouth opens slowly. Discrete movements are quick changes like a wink. Continuous movements are programmed to create gradual timbral changes while discrete movements are programmed for trigger-based control, such as starting and stopping a sample. In the following sections of the paper, the author will demonstrate the potential of the system for musical expression and present three examples of its functionalities. All three examples use different Csound and/or Cabbage instruments, allowing users to have control of more than one timbre at a time.

3 Sound Creation with Continuous Facial Movements

3.1 Controlling Frequency by Changing the Level of Mouth Opening

The following example focuses on the vertical opening of the mouth, as seen in actions like yawning or gasping. Game objects are attached to the upper and lower lips of the avatar. Changes in the user's lips position will be reflected accordingly (as shown in Figure 3a). *Transform.position* holds the values of a GameObject's transform parameters, which are the position, rotation and scale of an object. Since the action of opening one's mouth changes the Y position of corresponding game objects, Y position (*transform.position.y*) has been scaled and mapped to the frequency parameter of a Cabbage instrument, (as shown in Figure 2). This results in the user changing pitch in real-time while opening their mouth. This facial action is currently mapped to the frequency parameter to make the connection and control more obvious to the user, but it could also be mapped to control any other Csound channels in the Unity editor.

3.2 Adjusting Filter Cutoff Frequency and Reverb Size with a Smile

The following example uses the positions of mouth corners to adjust audio effects, focusing on sideway movements of the mouth. To increase expressive control of the system, two additional game objects have been added to the avatar's mouth on top of the pre-existing game object from the live capture package. The two game objects are attached to the mouth corners of the avatar. They measure the change in the mouth's sideway openings. For clarity in the explanation below, the game object on the left is referred to as Object A, and the one on the right as Object B.

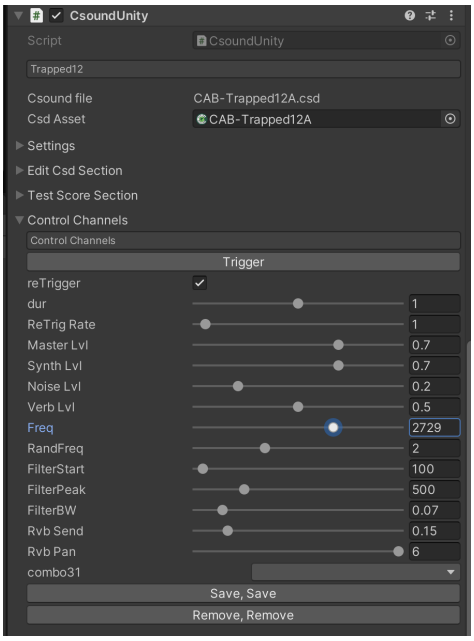


Fig. 2. The highlighted parameter is the frequency parameter, mapped to the Y position of the lips.

The X position of the two game objects corresponding to the mouth corners move in opposite directions as the user smiles. The X value for Object A decreases while that for Object B increases. Therefore, the mappings of channel values are inverse. Object A's position value is mapped to filter cutoff frequency inversely, so that when the X position value decreases, the cutoff frequency increases. Object B's position value is mapped to reverb size, which increases as the X position value increases. Consequently, as the user smiles, the cutoff frequency and reverb size increases, changing the timbre of sound in real-time.

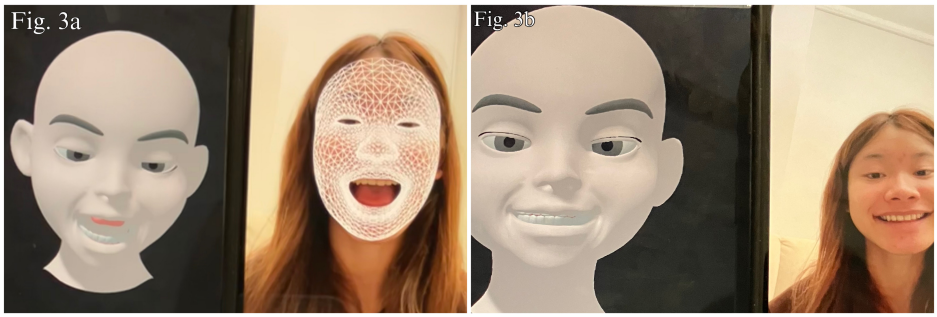


Fig. 3. The avatar mirrors the user's mouth gestures.

4 Sound Creation with Discrete Facial Movement

4.1 Triggering Drums with Eye Winks

This example demonstrates the system's functionality to trigger a drum beat by eye winks. In addition to the pre-existing game objects attached to the avatar's face mesh, four additional child GameObjects, two for each eye, have been added with a collider attached to each, in order to enhance accuracy in triggering instruments. The child GameObjects are added onto the top and lower eyelid of the avatar. One collider from each side has *Collider.isTrigger* enabled and contains a Rigidbody.

As blinking is a natural reflex movement, it requires more scaling when designing AR applications that interacts with such facial gestures. In this particular development, the system detects a rapid three-time collision between the two child GameObjects, meaning the eyes are blinked three times within a short period of time (currently set to within 2200 milliseconds). This reduces the possibility of system mistriggers, and proves that it was an intentional gesture by the user, instead of a natural reflex action.

5 Observations and Feedback from User-Testing Sessions

The first round of user testing sessions was conducted at MacLehose Medical Rehabilitation Centre.[4] Participants included stroke patients with limb motor impairment, patients with central nervous system diseases causing limb immobility, and patients with severe spinal cord injury, averaging around 50 years old.

All patients showed interest and excitement in the development of a new musical expression tool that they have never tried, and one that is different from their experience with familiar classical instruments. Some patients were particularly excited that this application required no prior musical training experience, suggesting they could jam with their caregivers using sonic textural layers without worrying about playing wrong notes. Some patients expressed initial confusion as they faced difficulties in controlling certain facial movements. Patients and healthcare professionals at the centre have identified and recommended facial movements, such as looking to the side and protruding the tongue, which people with physical conditions may find easier to perform consistently for facial detection.

User-testing sessions will continue to be conducted on a quarterly basis with system updates.

6 Future Plans

The author has built an improved version of the system now requiring only one mobile device with a front camera. More gestural interactions, such as eyebrow raises, have been designed for real-time musical expression. These new developments are subject to user testing in the next quarter. To enhance accessibility across platforms and resources, the author is also exploring the development of a browser-based system such that it could be run on a web browswer without a need for installation. More sound design capabilities will be implemented to the system, allowing the user to choose what parameter to adjust. The author is also exploring various face-detecting and face-tracking algorithms, such as but not limited to the MediaPipe Face Landmarker[9] and the Face Tracker[10]. Revisions to the system will continue based on feedback from user testing sessions, ensuring that the system is tailored to meet patients' needs.

References

1. Csound site, `https://csound.com/`
2. Cabbage Audio site, `https://cabbageaudio.com/`
3. Csound Manual site, `https://csound.com/manual.html`
4. Csound Unity, `rorywalsh.github.io/CsoundUnity/#/`
5. Unity - Scripting API, `docs.unity3d.com/ScriptReference/`
6. Unity - Live Capture, `https://docs.unity3d.com/Packages/com.unity.live-capture@4.0/manual/index.html`
7. Unity - ARKit XR Plugin, `https://docs.unity3d.com/Packages/com.unity.xr.arkit@4.1/manual/index.html`
8. Funk, Kuwabara, and Hiraga. Sonifications of Facial Actions for Musical Expression. In *Proceedings of the 2002 Conference on New Interfaces for Musical Expression (NIME-02)*, pp. 127-131.
9. Google AI Edge - MediaPipe Face Landmarker, `https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker`
10. Meta Spark - The Face Tracker `https://spark.meta.com/learn/articles/people-tracking/face-tracker/`

[4] MacLehose Medical Rehabilitation Centre is a rehabilitation centre in Hong Kong with specialties in neurosurgical rehabilitation, orthopaedics rehabilitation and geriatric rehabilitation.

Integrated Csound 2

Opening mind by opening architecture: analysis strategies

Francesco Vitucci[1] , Giuseppe Silvi[2] , Daniele Giuseppe Annese[3] ,
Francesco Scagliola[4] and Anthony Di Furia[5]

[1,2,3,4,5]Conservatorio "N. Piccinni" - Bari
[1]francescovitucci1@gmail.com
[5]anthonydifuria.sound@gmail.com

Abstract. In numerical signal processing for electroacoustic composition, the progressive loss of specific development and research environments caused by the increasing use of digital market tools has favoured the dominance of the closed-architecture audio processor model. This model, while powerful, envisions the possibility of describing output data about its perceived characteristics, but at the cost of ignoring its internal process and interacting systems, which become complex, powerful environments but closed in an inscrutable black box, a loss we must consider. Any digital signal processing technique tells a story. Just as the words of a language incorporate social, historical and technical polysemic layers, a signal processor has its own story of implementation, a gradual technological achievement with its inevitable aesthetic consequences. Through the looking-glass of literature, one can access those environments with renewed awareness by reestablishing a scientific method and an attitude to research. In this specific case, starting from the case study of Manfred Schroeder's historical reverbs, we illustrate the process of building analytical evaluation tools, as well as practical implementation, at the basis of a conscious study path.

Keywords: Reverberator, Faust, Porting, Impulse Response

1 Introduction

The process of IT democratization started with the introduction of the Personal Computer can be observed from multiple social perspectives. Unconditional accessibility to digital signal processing tools features a wide range of signal processors (DSP) to an ever wider range audience of users. This opening has generated a new category of producer users, who, fascinated by the ease of use and the speed of obtaining results, have become accustomed to the *black box* approach. This *closed architecture* model represents a paradigm in which users can use a process without necessarily understanding its internal workings: they evaluate the *output* based on perceived characteristics, ignoring implementation details, in one consequent social normalization in which the user, or the music producer, «è in realtà un *music prosumer* (che 'produce' solo consumando servizi e dispositivi tagliati su misura)»[1]. [4] Parallel to this trend is an ever-growing lack of specific research and development environments, in an extended sense of awareness and critical consciousness. The word *environment*, denotes «the conditions that you live or work in and the way that they influence how you feel or how effectively you can work»[2] and applies to the specific discussion, teaching or work environment, to design and implementation software, to large research centres, once the only environments where everything that is being discussed here was possible. Speaking of software, it is worth underlining that, the adoption of the model *black box*, with increasingly high-level function libraries, ready for use, cannot be inspected to understand how it works, promotes a superficiality in learning and using the tools sound processing.

In this scenario, training and education play a fundamental role in promoting an *open architecture* approach, encouraging students to explore and understand the theoretical and practical foundations of audio signal processing, in order to develop critical skills and creativity. This model, also called *white box*, [5] predicts that the focus is on understanding the underlying mechanisms of the process,

[1] «is, in reality, a *music prosumer* (who 'produces' only by consuming services and devices cut to size)», translated by authors.

[2] https://dictionary.cambridge.org/dictionary/english/environment

promoting critical awareness and in-depth understanding. The purpose of this discussion is therefore to show a path of understanding, acquisition and potential creative development. At the same time, we want to chart a course for future artistic and musical research works scientists who can find fertile ground in the use of the proposed paradigm to take root.

2 Analysis methodology

The implementation path that will follow is based on a series of specific choices. First of all, we chose to observe the reverberations from the point of view of the contents because the literature in this regard is open and accessible, in line with the proposed paradigm, as well as among the richest: precisely this richness has its violent counterpart inaccessible in prosumer use. It is precisely this method of use that, for example, has led to the use of convolution reverbs, the closed application of a mathematical system that does not contain the reverberation process but only imitates one of its results. On the contrary, historical algorithmic implementations have from time to time recombined basic reverberant components to obtain different results and, precisely in this process, a possible creative outlet was identified, in continuity with the attitude of the past: these were historical moments in which implementation primarily meant dealing with the limitations imposed by the machines available, thus generating an impulse for continuous investigation.

The starting point of the research was the implementation of M. R. Schroeder reverberator. In 1962, he developed the first digital reverb algorithm in history [6]. It is made of two basic components: a *delay in feedback loop*, or *comb filter* and an *all-pass filter*. The former is built up by inserting a simple delay line into a feedback loop; by doing so, one can produce multiple echoes, with exponential decay, as shown in Fig. 1. By mixing the direct sound and the delayed sound the *comb filter* is converted to an *all-pass filter*, a basic reverberant unit with flat frequency response.

The creation of this path was done in a textual programming environment, a choice attributable to educational needs, as the understanding of all the pieces of classical programming is more direct: declarations of variables, the definition of functions and their concatenation. In this specific case, we initially chose to use the *Faust* language (*Functional Audio Stream*) which is a functional programming language for sound synthesis and audio processing.[3] In addition to being textual, this language has a series of specific characteristics of an open architecture model: the highest level functions available are implemented using the same language, allowing the operation to be observed in every greater depth, up to the basic "primitive" components; the tools provided have analytical capabilities that allow both to visualize the sample-by-sample behaviour of a processor and to generate its block diagram. This represents the background of the "Reverberators" project[4] [14], a path of reconstruction and conscious implementation of reverberators from historical literature.

The idea therefore of porting the project into a Csound[5] [1] environment, in an attempt to preserve the chosen methodology, was immediately measured with the ability to build precise analysis tools for the impulse responses of the components under construction (Sec. 3). However, it is not simply a question of replacing a language with an alternative, but rather the integration of the various experiences, merging the strengths of the various approaches: from this perspective it is possible to observe the reconstruction of some of Faust's composition structures, such as `par` and `seq`[6], indispensable for the implementation path (Sec. 4). Precisely in this step, we encountered different efficiency limitations between the two languages. Therefore, understanding that these are teaching study strategies, we want to underline how the construction of compiled csound opcodes in C is the next step which, in the growth of the project, will represent a future stage of development.

3 Impulse Response

As already explored in other research works [8], once a component has been developed, it is necessary to test its impulse response (*IR*). To create a precise *IR* analysis routine, an appropriate orchestra

[3] `https://faust.grame.fr/`
[4] `https://github.com/s-e-a-m/faust-libraries/blob/master/src/seam.schroeder.lib`
[5] `https://github.com/s-e-a-m/csound-libraries`
[6] `https://faustdoc.grame.fr/manual/syntax/`

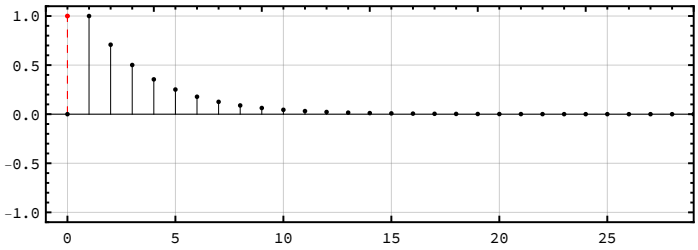


Fig. 1. Plot of the impulse response of the `FDL` component, obtained from within Csound with the proposed analysis routine.

of two instruments was developed, as can be seen in the code snippet below. `instr 1` is responsible for generating a Dirac pulse, testing the component (Schroeder `FDL`[7]), writing the *IR* in a globally accessible table and scheduling `instr 2`. The latter plots the values in the console and a txt file; then it can be parsed to extract the desired amplitude values (for this purpose a script in Wolfram Language[8] was used, which also automates the graphic representation process (Fig. 1).

Analysis routine and plot of the impulse response

```
<CsoundSynthesizer>
<CsOptions>
-n -d -+rtmidi=NULL -M0 -m0d
</CsOptions>
<CsInstruments>
ksmps = 32
nchnls = 2
0dbfs = 1
#include "schroeder.udo"
instr 1
  setksmps 1
  ires system_i 1, "mkdir -p PLOT"
  iT = 1;time
  iG = 1/ sqrt(2);gain
  iSR = sr;sample rate
  aDirac mpulse 1,1
  aR = 0
  iLenTable = 32
  giFDL_PLOT ftgen 2, 0, iLenTable, -2, 0
  aFDL_PLOT = FDL_SCH(aDirac, iT, iG, iSR);process to test
  aIndex phasor sr/iLenTable
  tablew aFDL_PLOT, aIndex * iLenTable, giFDL_PLOT
  schedule 2, iLenTable/sr, 1
endin
instr 2
  ftprint giFDL_PLOT
  ftsave "PLOT/1FDL_PLOT.txt", 1, giFDL_PLOT
  ires system_i 1, "wolframscript -script plot.wls"
  exitnow
endin
</CsInstruments>
<CsScore>
```

[7] The `FDL_SCH` and `APF_SCH` UDOs can be consulted at the following link: `https://github.com/s-e-a-m/csound-libraries/blob/main/src/schroeder.udo`

[8] `https://www.wolfram.com/language/`

```
i1 0 100
</CsScore>
</CsoundSynthesizer>
```

4 Compositional Structures: `par` and `seq`

Once the evaluation and analysis tools have been acquired and the actual effectiveness of the constructed basic components has been tested, the blocks can be composed. If we observe Schroeder's reverb implementation model (Fig. 2), we identify two behaviours that deserve attention: a first construct with structurally identical blocks (but with different parameters) positioned in parallel precedes a second in which the blocks are chained in series.

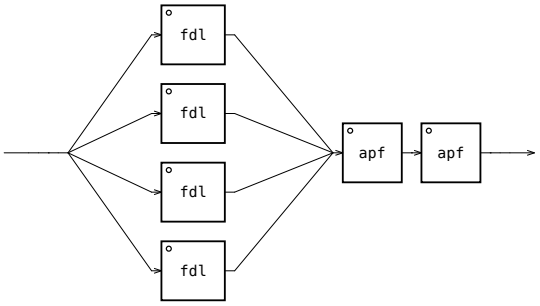


Fig. 2. Faust Block diagram of Schroeder's reverb implementation: a section of four *comb filters* in parallel precedes a chain of two *all-pass filters* in series.

These two compositional structures are already supported by Faust (see Sec. 2), but are not present in canonical Csound language, so two *User Defined Opcodes* were specifically written (as can be seen in the code snippet below). At the moment the proposed UDOs are not generic, as Faust `par` and `seq`; nevertheless, they are easily adaptable for any other opcode to compose in parallel and in series.

***par** and **seq** User Defined Opcodes*

```
;--------PAR FDL-----------------------------------------------------
opcode PAR_FDL_SCH, a, ai[]i[]iii
  aPulse, iTfdl[], iGfdl[], iSR, iN, icnt  xin
 icnt  =  icnt + 1
 aFDL = FDL_SCH(aPulse, iTfdl[icnt - 1], iGfdl[icnt - 1], iSR)
 aMix init 0
 if icnt < iN then
   aMix PAR_FDL_SCH aPulse, iTfdl, iGfdl, iSR, iN, icnt
 endif
 xout aMix + aFDL
endop
;--------SEQ APF-----------------------------------------------------
opcode SEQ_APF_SCH, a, ai[]i[]iii
  aPulse, iTapf[], iGapf[], iSR, iN, icnt  xin
 icnt  =  icnt + 1
 aAPF = APF_SCH(aPulse , iTapf[icnt - 1], iGapf[icnt - 1], iSR)
 if icnt < iN then
   aAPF SEQ_APF_SCH aAPF, iTapf, iGapf, iSR, iN, icnt
 endif
 xout aAPF
endop
```

Of course, a simple chain like the one shown in Fig. 2 can also be created without the illustrated UDOs. However, it becomes difficult to ignore them if the number of components in series or parallel becomes much higher, to research and study the creative expansion of historical processes, one of the topics underlying the "Reverberators" project. [8]

5 Conclusions

It is appropriate to underline the strong didactic aspects of the proposed paradigm. Opening architectures means opening paths of understanding and learning; it means colliding with and circumventing external problems and limitations; it therefore means not *consuming* but *producing* the environment with which to interact, in a virtuous mechanism of musical, creative and professional growth.

Schroeder's work [6] clearly shows architectures impulse responses that led the first Faust implementation. [8] Its Csound porting [1] produced an analysis environment, necessary for the understanding and consequent conscious writing of open architectures and further creative applications.

References

1. Accessed: 2024/07/29. `https://github.com/s-e-a-m/csound-libraries`.
2. Lazzarini, V. et al.: Csound: A Sound and Music Computing System. Springer (2016)
3. J. Heintz, *Floss Manual Csound*, 2023. Accessed: 2024/07/29. `https://flossmanual.csound.com`.
4. A. Di Scipio, *Circuiti del Tempo*, p. 560. Libreria Musicale Italiana srl, 2021.
5. M. E. Khan and F. Khan, "A comparative study of white box, black box and grey box testing techniques", *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 6, 2012.
6. M. R. Schroeder, "Natural sounding artificial reverberation", *J. Audio Eng. Soc*, vol. 10, no. 3, pp. 219–223, 1962.
7. M. R. Schroeder and B. F. Logan, "colorless artificial reverberation", *IRE Transactions on Audio*, no. 6, pp. 209–214, 1961.
8. D. G. Annese, F. Vitucci, A. Di Furia, F. Scagliola, and G. Silvi, "Archeotopologie: implementazione critica di memorie senza colore," *Atti del XXIV Colloquio di Informatica Musicale*, 2024.
9. M. R. Schroeder, "New method of measuring reverberation time", *Acoustical Society of America*, 1964.
10. M. R. Schroeder, "Digital simulation of sound transmission in reverberant spaces", *The Journal of the acoustical society of america*, vol. 47, no. 2A, pp. 424–431, 1970.
11. C. Roads, *The computer music tutorial*. MIT press, 1996.
12. Accessed: 2024/07/29. `https://ccrma.stanford.edu/~dattorro/Griesinger.pdf`.
13. Accessed: 2024/07/29. `https://ccrma.stanford.edu/~dattorro/Griesinger.jpg`.
14. Accessed: 2024/07/29. `https://github.com/s-e-a-m/faust-libraries/blob/master/src/seam.schroeder.lib`
15. J. Dattorro, "Effect design, part 1: Reverberator and other filters", *J. Audio Eng. Soc*, vol. 45, no. 9, pp. 660–684, 1997.
16. W. G. Gardner, "A realtime multichannel room simulator", *J. Acoust. Soc. Am*, vol. 92, no. 4, p. 2395, 1992.
17. W. C. Sabine, *Collected papers on acoustics*. Harvard university press, 1922.
18. R. Vermeulen, "Stereo-reverberation", *J. Audio Eng. Soc*, vol. 6, no. 2, pp. 124–130, 1958.
19. M. R. Schroeder, "Listening with two ears", *Music Perception: An Interdisciplinary Journal*, vol. 10, no. 3, pp. 255–280, 1993.
20. J. O. Smith, "Physical audio signal processing", 2010, Accessed: 2024/05/07. `https://ccrma.stanford.edu/~jos/pasp/Freeverb.html`.
21. J. A. Moorer, "About this reverberation business", *Computer music journal*, pp. 13–28, 1979.

Integrating Csound into Unreal Engine for Enhanced Game Audio

Albert Madrenys Planas*

Maynooth University
amadrenys@gmail.com

Abstract. Unreal Engine is one of the most widely used game engines in the current market, thanks to its exceptional flexibility and strong graphical capabilities. Recently, the development team has introduced a new tool called MetaSounds, designed to facilitate sound synthesis, digital processing and sound design in a native way and within a node-based interface. Despite its user-friendly interface, MetaSounds still lacks certain functionalities present in older sound engines such as Csound or SuperCollider. Currently, integrating Csound into Unreal needs the use of a middleware like FMOD or Wwise, along with Cabbage to export Csound code into a VST. However, a MetaSounds node that inherently incorporates Csound, without the use of external dependencies, and with MetaSounds adaptable, intuitive, and potent graphical interface would be a significant advancement. Thanks to Unreal Engine's support for C++ implementations and enabling developers to craft their own MetaSounds nodes, it can be possible to integrate Csound within a MetaSounds node through the Csound C++ API.

Keywords: Unreal Engine, game audio, video game, game engine, MetaSounds, Csound, plugin, MetaCsound, adaptative audio, adaptative music, C++

1 Introduction

With the gaming industry expanding rapidly, it is not surprising that studios are constantly pushing boundaries once thought impossible. In the last decade, significant effort has been put into audio, specifically with realistic spatialization and adaptive audio, both in real-time. Adaptive audio is described as audio or music whose characteristics change in response to events in the game.

Game engines are one of the main tools in game development. They provide a set of commonly used tools in video games, such as rendering, physics, sound, and graphical user interfaces, all optimized for real-time execution. Several solutions have been created to allow adaptive audio in these engines. Developers usually rely on multiplatform middleware such as FMOD or Wwise, which can be integrated into the project or engine, although integration solutions have already been made for the most popular engines.

As Csound already has many features that are very useful for adaptive audio and music, this work focuses on creating a tool that enables Csound inside Unreal Engine.

2 Unreal Engine and Metasounds

Unreal Engine, developed by Epic Games, is one of the most popular engines at the moment. Unreal has established itself as an industry standard in areas such as realistic rendering and audio. Although it is coded in C++ and allows developers to code games in C++, it also supports the use of blueprints, a node-based programming interface. This makes the engine accessible to people who are not specialized in programming. Furthermore, it is free to use from the start and only charges developers once the game starts generating revenue.

Recently, Epic Games has released a new feature in their engine called MetaSounds. Currently in beta, MetaSounds is a high-performance audio system that provides audio designers with complete control over a digital signal processing graph to generate sound sources. Similar to Unreal's blueprints, it uses a node-based interface, making the connections between processing units, such as oscillators

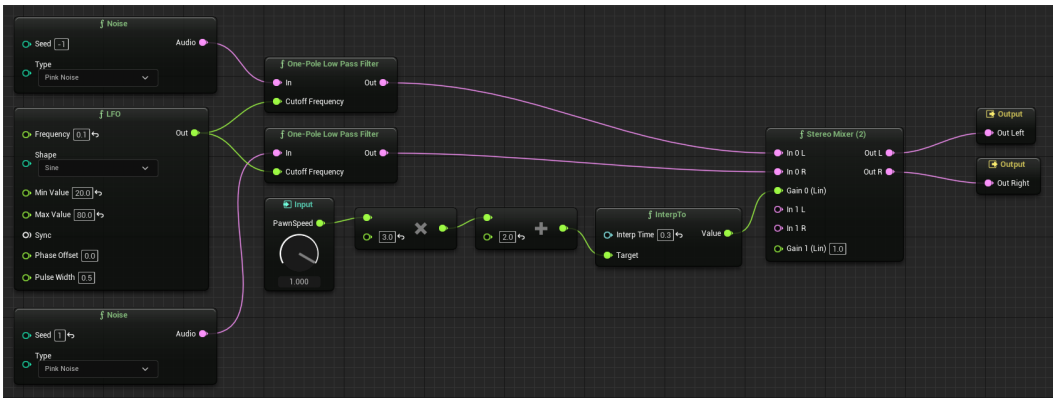


Fig. 1. MetaSounds graphic interface.

and filters, very easy and approachable. MetaSounds allows developers to perform complex digital signal processing (DSP) operations without the need for middlewares.

In a typical MetaSounds graph, the input nodes are created on the left side of the graph. These inputs can be graph variables that are written by other blueprints, making it easy to map them to events happening in the game, such as the player moving, a new enemy appearing, or a car passing by. In the center of the graph, the DSP nodes are placed, interconnected with each other, the graph inputs, and the graph variables. On the right side, the outputs of the graph are placed. Usually, there will be one audio output channel if the graph is meant for mono audio, or two output channels for stereo audio. An example of a MetaSounds graph is shown in Figure 1.

Despite the strong foundations and potential that MetaSounds has, it is still a very young environment and lacks many functionalities. While you can perform basic tasks such as adding a low-pass filter or pitch shift, their capabilities are relatively minor compared to older audio environments. However, if we could create a new MetaSounds node that runs an instance of Csound inside, receiving audio and control-rate inputs from other MetaSounds nodes and allowing outputs to be sent to other nodes, we could provide developers with far more options.

Despite the strong foundations and potential of MetaSounds, it is still a very young environment and lacks many functionalities. While it supports basic tasks like adding a low-pass filter or pitch shift, its capabilities are limited compared to more established audio environments. However, by creating a new MetaSounds node that runs an instance of Csound, receiving audio and control-rate inputs from other MetaSounds nodes and allowing outputs to be sent to other nodes, the available options could be significantly expanded.

3 MetaCsound

The scope of this project is to create a new plugin for Unreal Engine, specifically for the MetaSounds module. The plugin, called MetaCsound, aims to integrate Csound into the MetaSounds. This is achieved by adding a new family of nodes. Since MetaSounds allows developers to create their own nodes in C++, and Csound offers a C++ API, integration is possible.

This family of nodes will have different vertex interfaces but will function the same at their core.

3.1 Vertex Interface

The vertex interface refers to the input and output pins in MetaSounds that can be interconnected between nodes. In terms of Csound, these will correspond to audio channels and control busses.

Additional pins will be needed. The first one will specify the name of the Csound file to be executed in the node. This file must be placed in a designated folder within the Unreal project. There will also be play and stop pins. The play trigger input pin will initiate the compilation and execution of the

* The author would like to thank Victor Lazzarini for his supervision and guidance.

Csound file, while the stop trigger will halt the performance. The play input can be triggered multiple times to restart the score. Finally, a finished trigger output pin will be included. This output pin will be activated when the node finishes performing, either because the Csound score has concluded or the stop input has been triggered.

3.2 Node behaviour

When the node starts running inside the Unreal Environment, a new Csound instance will be created thanks to the Csound C++ API, and it will compile the file passed by reference. During each MetaSounds buffer call, the audio and control inputs will be read and written to the Csound input channels and buses accordingly. Then, the audio output channels and the control output buses from Csound will be read and written to the MetaSounds output vertexes. Since MetaSounds and Csound may work with different buffer sizes, the Csound call to perform the next block will only be triggered by MetaSounds when the node detects that the buffer has been emptied and requires new data.

One key thing to note is that in order to ensure satisfactory integration between MetaSounds and Csound, they both have to operate with the same samplerate. Therefore, the samplerate of Csound will be overridden to match the samplerate of MetaSounds.

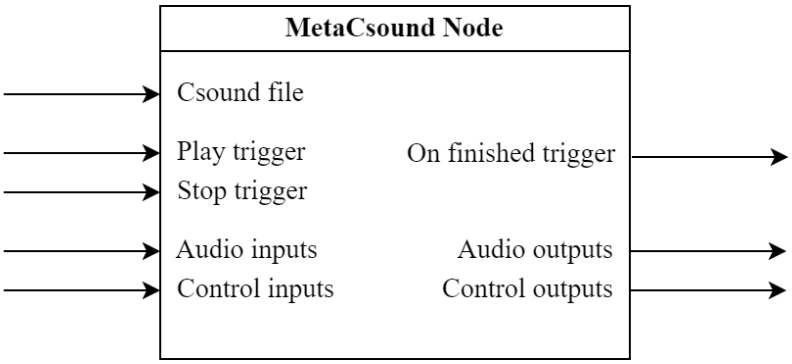


Fig. 2. New MetaCsound node layout.

3.3 Design iterations

When initially designing the project, the plan included a dynamic vertex interface. This involved compiling a new Csound file when detected to scan for every audio channel and control bus and then mapping them to MetaSound's respective vertex equivalents in the vertex interface. However, at the time of writing this paper, implementing this feature seems very challenging due to MetaSounds being in beta and the documentation being incomplete. To address this limitation, different variations of the node will be created, each with a different static vertex interface. This will allow developers to choose the version of the node that best suits their needs. However, this approach will require the control buses of the Csound file to be named with specific names so that the MetaSound node can map the control bus and the vertex accordingly. The layout of the final design is shown in Figure 2.

4 Discussion

In the example shown in Figure 3, noise and saw wave audio signals are sent into the Csound node. Csound will modify the output depending on the opcodes used in the Csound file with the name specified by the *File* input. The node will consider the two control-rate inputs to modify the audio signals. It will start and stop performace according to the corresponding triggers. When the node stops, the *On Finished* trigger will be activated, causing the *Wave Player* node begin performing.

The impact of this project could be significant on future game development. Traditionally, adaptive audio in games has been handled by middleware such as FMOD or Wwise. While these middleware are powerful, they consume a lot of resources, and developers have to pay fees when the shipped

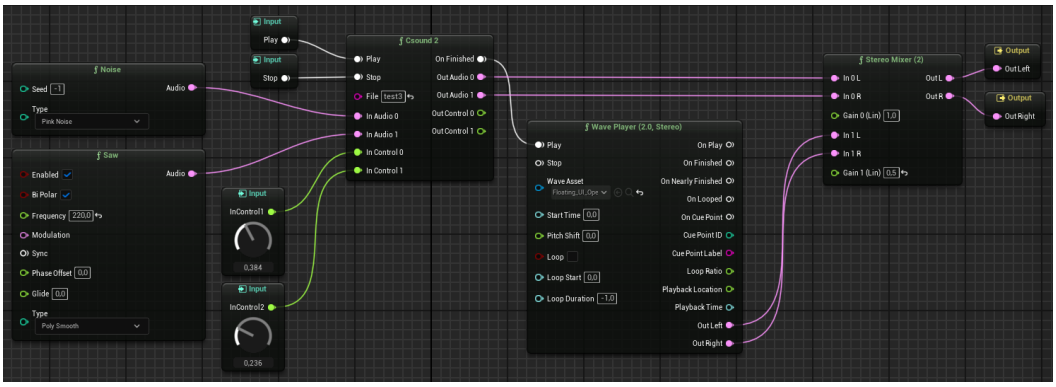


Fig. 3. Example of a MetasSounds graph using the new Csound node.

game surpasses certain revenue thresholds. Additionally, integrating middleware into the engine can be tricky at times. Since MetaSounds is a module of Unreal, the integration is already done, requiring fewer resources and less work.

Prior to this project, attempting to utilize the capabilities of Csound inside Unreal required the use of one of the previously mentioned middleware. Developers would also have to import a VST created with Cabbage, containing the Csound code. For Unity Engine, the other most popular engine besides Unreal, a less convoluted option already exists, CsoundUnity, which allows Csound to be run inside the engine directly without the need for middleware and VST exports through Cabbage. Unreal Engine lacked a similar tool. With MetaCsound, Csound code can be easily run inside the MetaSounds module in a far more straightforward and user-friendly manner, making this project highly useful and relevant for both the Unreal and Csound communities.

5 Conclusions

The project has been successfully implemented. The current nodes are operational, robust, and easy to understand. They include a Csound file passed as a string, multiple audio input and output channels, multiple control-rate input and output channels, a start and stop input trigger, and a finished output trigger (see Fig 3). Despite these successes, there are additional features that could be implemented in the near future. These include a better method for sending the Csound file, making the plugin available for macOS and Linux, and integrating both MIDI messages and score events. The project is already demonstrating its capabilities and great potential.

Hopefully, as MetaSounds establishes itself as a reliable tool and exits its current beta state, more developers will begin to use it, and the use of more convoluted paths such as middleware will be reconsidered for certain projects. If that is the case, MetaCsound could become a key tool for integrating complex DSP operations into a MetaSounds graph, making Csound one of the best audio programming options for adaptive sound in Unreal Engine.

The source code of the project can be found in the following GitHub repository: `https://github.com/AlbertMadrenys/MetaCsoundProject/`

References

1. Lazzarini, V., Yi, S., ffitch, J., Heintz, J., Brandtsegg, Ø., McCurdy, I.: Csound: A Sound and Music Computing System. Springer (2016)
2. Csound API documentation site, `https://csound.com/docs/api/index.html`
3. Epic Games: MetaSounds on Unreal Engine — Unreal Engine 5.1 Documentation, `https://docs.unrealengine.com/5.1/en-US/metasounds-in-unreal-engine`
4. Firelight Technologies: FMOD — FMOD for Unreal, `https://www.fmod.com/docs/2.02/unreal/welcome.html`
5. Walsh R.: CsoundUnity documentation site, `https://rorywalsh.github.io/CsoundUnity/`

The advantages of multi-dimensional interfaces for the future of Csound

Hans Pelleboer

Perceptual Engineering
hanspelleboer@online.nl

Abstract. Present day microcontrollers allow many physical properties to be translated to and from the digital domain. As non-trivial sound synthesis encompasses a large number of controlling variables, the necessary properties of an effective interface are discussed. The dichotomy between the analytic approach of computer-mediated electroacoustics and Gestalt-based integrated human perception is shown. Special emphasis is laid on the importance of simultaneous multi-modal presentation for sensory integration and the vital role played by haptic and proprioceptic feedback. Comparisons are made between the established conventions of analog electronic equipment and the relative pioneering status of computer synthesis. Two interface designs are presented, illustrating practical steps on the possible path forward and the implications these would have for Csound's further development.

Keywords: Microcontrollers Ergonomics Neurosensory Integration

1 Introduction

In the dark ages, when digital computing power was very limited, the only way to realize sound synthesis was through batch processing. Nowadays, the classic cycle of compilation, playback and editing is still prominent. Yet, a strange discrepancy has emerged: the Csound variable PMAX, which sets the maximum number of arguments on a score line, has grown to 1000 over the years [1]. Elementary combinatorics show that a thousand-dimensional parameter space presents a complexity that can never be explored by editing or simple algorithms; life is simply too short and musical memory too brittle -- a much more direct approach with simultaneous access to a large number of parameters in real-time is required to break down this task into manageable chunks.

Analog music equipment with lots of knobs and faders enjoys its immediate, intuitive appeal for good reason: Direct access to all vital variables, settings can be read at a glance, there is often a one-to-one mapping between a parameter and its corresponding knob, while real-time control gives immediate feedback which enables you to tweak all settings to get them `just right'. The spatial layout of a well-designed control panel will become embedded in the muscle memory of the operator, thereby facilitating rapid, yet precise, interaction. Is it any wonder people just can't keep their hands off them?

And there is more: Whereas an experienced conductor can read a 40-stave score page at a glance, the hapless Csounder has to decipher pages and pages of cryptic symbols to achieve the same. Being able to synthesize sound with a fine degree of control over all elements involved presents an immense opportunity, but it comes with a catch. As Gottfried Michael Koenig already observed in 1955: "Electronic music is the final stage of the process in which the continuum of natural sound -- first dissolved in isolated musical instruments -- falls apart in isolated parameters; the continued atomization falls back into undistinguishedness [2]." Human perception is completely at odds with this approach: We do not observe loose notes, let alone atomized parameters, but the Gestalt of a

complete musical phrase. Our senses do not operate very well in isolation; all our sensory modalities, while simultaneously processing input, have to come to some sort of an agreement in order to make sense of our world [3]. This integration reaches its pinnacle in the coordinated movements involved in playing a musical instrument; the positioning of limbs and their higher derivatives; movement, acceleration and jerk, the precisely measured generation of force, visual and auditory feedback, all come together in a whirlwind of independent operators acting together in perfect harmony.
Compared to this, your typical computer music interaction is extremely limited; Keyboard strokes are one-dimensional, graphical interfaces that are mouse driven, even elaborate ones, control one parameter at a time, while all others remain invariant, the environment remains atomized. I think that much more sophisticated interfaces are needed to break through this barrier. Before all, the ability to control Csound while no longer bound to your desk is a necessary step to enter the realm of performance. A short review of what has been achieved thus far.

2 Historical Interfaces

Electronic sound generation, lacking historic precedent, promised to make a clean break with all existing musical conventions. In reality, things turned out quite differently. Apart from some outliers -- the theremin comes to mind -- most inventors played it safe and chose familiar interfaces like the piano keyboard. The consequences of such a choice are twofold; first of all, you present something accessible that users understand rightaway, which is commercially advantageous. But the price that you pay is that the provocative sharp edge of the completely new is immediately blunted; all the downtrodden paths of conventional piano fingering lead inevitably to conceptual impoverishment. Indeed, it is no coincidence that people like Bob Moog and Don Buchla vehemently opposed anything resembling a piano keyboard attached to their synthesizers as that would sterilize countless seeds of original musical thought. Developing interfaces to other classes of instruments proved very difficult; the ARP corporation bankrupted itself while working on the Avatar guitar interface.

Nowadays, our situation is very different; nearly every physical variable imaginable can be processed by cheap microcontrollers and their numerous peripherals. As development will focus primarily on optimizing ergonomics, economic considerations that force low knobs-per-dollar ratios are not really a matter of concern. In fact, they should be avoided at all cost: having to wade through multi-level submenus while performing is downright disastrous. With that matter settled, let us discuss what makes an effective user interface.

2.1 The three Elements of an Interface

An interface provides the user with three functions that will, for simplicity's sake, be named Read, Play and Feedback.

Read: the ability to evaluate a variable by simply looking at the controller, like an organ stop or a multiturn dial. This a relatively static category, in which precise control carries more weight than rapid intervention. Rotary knobs, rocker switches and the aligned set of faders on a graphic equalizer all represent the read function.

Play: those controllers that provide direct, fast access that is primarily haptic and motoric in nature. A dancer fitted with electromyographic electrodes to all major muscle groups represents the most direct form of the play category. Play can be enhanced even more by parallellism: a small portable Bluetooth device linked to a single dancer will see its function potentiated mightily when

used by the entire corps de ballet; the whole is *much* larger than the sum of its parts.

Feedback: all elements that report back the actual status of the process being controlled. In acoustical instruments, tactile feedback is ubiquitous; the sudden change in lip pressure on the mouthpiece of a brass instrument when one jumps from one harmonic to another, for example. On the electronic domain, feedback is predominantly visual; one can think of blinking leds in LFO's, sequencers and envelope followers, or a small display that shows the value of a variable, allocated to a corresponding control knob. Unfortunately, tactile interfacing with the computer is severely underrepresented. Computergamers are clearly ahead of us here; control sticks that kick back, steering wheels and brake pedals that generate proportional counterforces, thus mimicking real automotive action, are well-established. If only computer music instruments would get *that* physical!

One should note that these three functions are conceptual abstractions, real world components will nearly always encompass elements of more than one. A large, smooth running fader, for example, has a clear 'Play' function, but its position can also be read visually, which puts it in the 'Read' category.

An elegant example of a device that comprises all three functions is a mixing console: It consists of a Read section; the arrowed knobs for equalization, channel gain and panning, a Play section with large, easily accessible faders right in front for rapid gain-riding, and the metering bridge at the rear which provides visual Feedback of what is actually happening.

2.2 The Well-tempered Interface

Given the ‘open’ nature of Csound, a static one-size-fits-all interface is not really practical. Also, over-specialization makes rigid, which inhibits functionality. Just like snippets of code that can be combined at will, interfacing functions should be available on demand and easy to combine. This is very difficult to realize in practice; there is a lot of mechanical thought and work involved in designing an ergonomic interface. Linking mechanical functions to electronic processing presents its own set of problems; between the slightest perceived perturbation of hair follicles and the strongest instrument grip of our hands lay nearly six orders of magnitude of force; there is no transducer available that covers that dynamic range. Nevertheless, there are many building elements available that can detect position, movement, vibration, warmth. All can be used to reach a high degree of sensory integration: the more modalities involved, the more the action will `stick' in the experience of the user. Like the mixing console already mentioned, most succesful interfaces have gradually evolved over time instead of appearing *de novo* in their definite form. The best way to realize versatile, yet powerful, interfaces is to modularize: Develop small interfaces with optimized ergonomics and combine these at will.

Some examples:

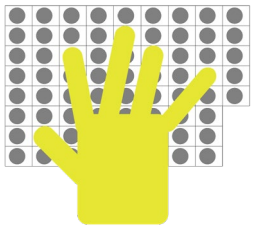


Fig. 1. STM32 Joystick Controller.

Maximum packing density is achieved with 69 cheap PS2 joysticks, fed into a single STM32 Blue Pill [4]. This 32bit ARM microcontroller features dual I2C buses that can be fitted with 16 ADC chips and 8 I/O expanders. Combined with its native ADC inputs, it gives access to 207 dimensions of control within the reach of a single hand. The square eight-by-eight matrix plus five in a row is a deliberate choice; minimal distance to traverse and no rows of twelve that nudge you towards the chromatic octave.

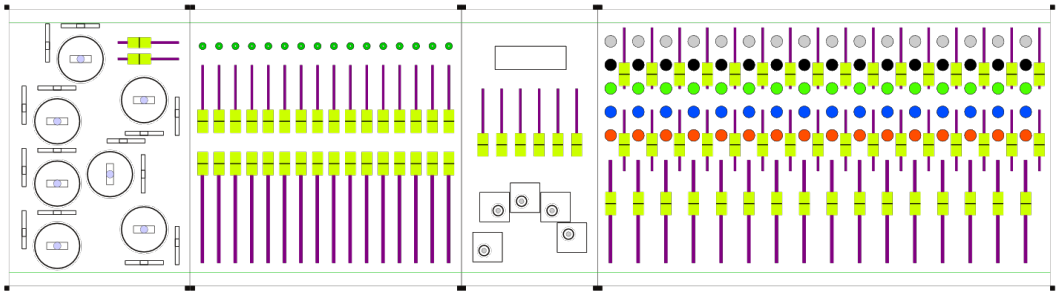


Fig. 2. The Village Square

This is the approach that I consider the most promising: specialized small modules, each centered around a single microcontroller, to be combined as needed. The leftmost module consists of two sets of joysticks; pure 'Play'. Next to that is a Read interface with large faders. The module in the center combines a hand of joysticks with preset faders and a small display for Feedback. The module on the right provides the familiar ergonomics of a mixing console.

3 Integration into Csound

Electronic sound generators have been with us for more than a hundred years. Yet the very fact that we are still discussing their application, proves that their position is far from established: There is no such thing as a recognized virtuoso school of players, taught as part of the regular curriculum at conservatories worldwide. I do not pretend to know the definite answer to this riddle, but feel very strongly that a more direct, more tangible, Man-Machine interface is a necessary part of the solution. Historically, software development has lagged behind hardware innovation and this time is no exception. As these examples show, mainstream microcontrollers can be combined with off-the-shelf hardware to generate massive streams of control data. Problem is that Csound in its present state cannot accommodate peripherals on this scale: The microcontroller related opcodes allow for only one device connected at the time, with unidirectional data flow, thereby excluding feedback of any sort. The maximum 'bandwidth' of thirty simultaneous data threads, set by the variable MAXSENSORS in serial.c, is more than an order of magnitude smaller than what even these relatively simple circuits are already able to fill [5]. Extending the bandwidth to hundreds of threads might require drastic adaptation of Csound's infrastructure.

Perhaps this situation presents the first inkling of a paradigm shift in the Kungian sense; given that MusicN type languages have been around for some sixty years, this may prove inevitable. The Csound community is larger than ever, adaptation and growth are hallmarks of any vital entity, so we can be confident that these new developments will also become well-integrated in due time.

References

[1] The Canonical Csound Manual Version 6.17.0
[2] G. M. Koenig 1955 Die Reihe I, Universal Edition, Wien
[3] Gestalt Psychology, Encyclopaedia Brittanica
[4] https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html
[5] csound-6.18.1/Opcodes/serial.c

KEYNOTES

Frippertronics

Victor Lazzarini

Music Department, Maynooth University Ireland
victor.lazzarini@mu.ie

Abstract. This paper describes the main ideas around the performance of Frippertronics at the ISCS 2024. The principles of the genre and the methods used are introduced. This is followed by an exploration of the design of the instrument used in the performance. A brief discussion of the musical approach complements the paper.

Keywords: delays, feedback, improvisation
Keynote video recording: https://doi.org/10.21939/icsc2024-Lazzarini

1 Introduction

I call *Frippertronics* a genre of musical performance based on the use of long delay lines with feedback. The term was introduced by Robert Fripp to designate his use of tape-based delays in an improvisational context, which was first shown to him by Brian Eno and resulted in the recording of *No Pussyfooting* [1]. Following this, he developed his own individual approach and musical aesthetics that took advantage of the soundscapes created by the interaction of sustained guitar sounds layered onto tape. From my perspective, his use of simple sound processing technology, coupled with elements of a sonic language, characterises a specific genre that is open for exploration by other musicians. Fripp himself indicated that it could be thought of a contemporary counterpart of a classic chamber ensemble such as the string quartet or the wind quintet. So, taking one step further, I started developing performances simply named as *Frippertronics*, in a similar fashion to how I would have written a *string quartet* or a piano *sonata*. Thus in 2023 I released an album [2] containing such a work in two parts, as a record of my exploration of the genre. The opportunity to perform at the ICSC allowed me to continue to develop this work.

2 From Tape to Csound

The original approach taken by Fripp, from Eno, was to use two tape recorders sharing a single tape. The first tape recorder recorded onto it and the second was used to play it back, feeding back the signal into the first. Similar types of tape arrangement had been used before in electronic music since the 1950s and 1960s, and this was by no means unprecedented. It also formed the base of standard echo effects used by studios of the time. Its use with a heavily saturated and sustained guitar sound was probably one of the unique elements brought to the table by Fripp, as well as the musical language developed around the setup.

From this starting point, I sought to develop a simple simulation of the process using Csound [3] as the programming/performance environment. I could only envisage an approximation of the original analogue system, and my approach was in no way to attempt a faithful reproduction of it. There were far too many open questions about how the tape recorders were connected together, the tapes used, the feedback levels, etc. However, I was very interested in building a system that could be played with, learned, reconfigured, and that supported expressive performances, all of this based on the original principles of the tape delay system. Csound was an ideal choice for me since it had all the building blocks ready for me to put together an working instrument.

2.1 Program Design & Code

I started with the requirement of a long delay line (minimum of around 7 secs, which seemed to be around the length used in the tape systems), and feedback. One of the points of an analogue feedback arrangement is that it will probably not have a flat frequency response, and so high frequencies in

particular may get degraded over time. To emulate this, I placed a first order lowpass filter in the feedback loop.

Noise would possibly build up as well, and maybe amplitude distortion in the form of tape saturation, but I decided to leave these aspects out of the system. To me, it was essential to lose high frequencies over time, which could be made to have quite an expressive effect ("sonic memories fading away"). With this simple arrangement, I had a basic instrument to explore some initial performance ideas. It was very clear from this, that control over cutoff frequency was an essential requirement. Also, having both lowpass and highpass filters seemed to be a more complete approach.

One advantage of a desktop processing environment is that we have very few limitations in terms of memory and so it is possible to add other delay lines if we wish. The question becomes one of how to manage the complexity of the system. With this in mind, I started to experiment with a double delay arrangement. The final form of the instrument then included a two-channel input, two delay line lines with feedback and LP-HP filters in the signal path. Controls were provided for cutoff frequencies and a switch to turn audio input on/off. To complement the program, some spatial audio processing was incorporated in the form of stereo panning (at the instrument output) and chorus/reverb (as a global process). This is shown in the flowchart of Fig. 1.

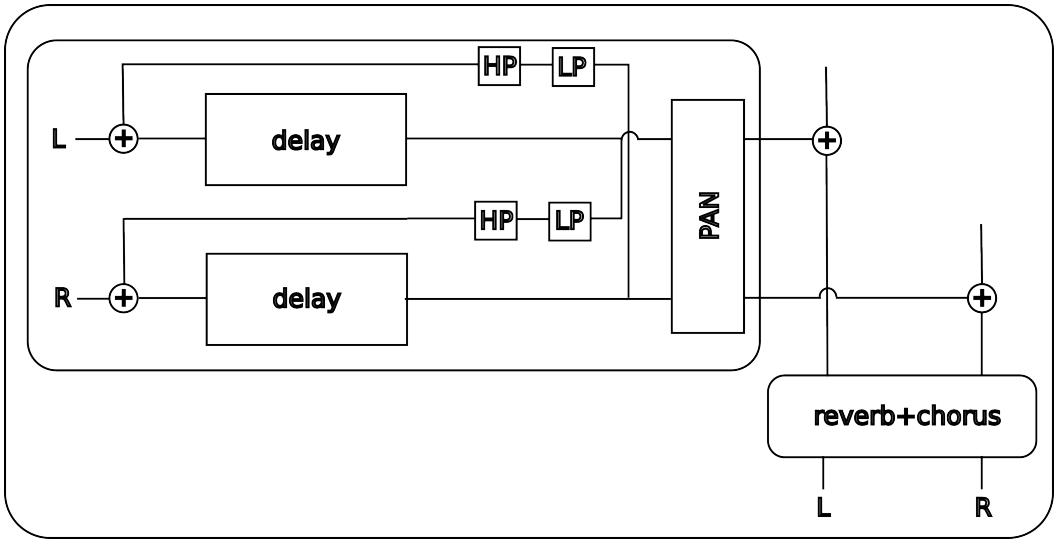


Fig. 1. Program Flowchart.

This design was then implemented in Csound (version 7.0 [4]). Control input is provided by bus channels, which may come from a host, or in this case via Open Sound Control (OSC) messages sent to the server (open to input on port 7000).

```
<CsoundSynthesizer>
<CsOptions>
-odac -iadc -dm0 -port=7000
</CsOptions>
<CsInstruments>
0dbfs = 1
nchnls = 2

;;channels
chn_k "hpf",3
chn_k "fdb",3
chn_k "lpf",3
chn_a "left", 3
chn_a "right", 3
```

```
instr 1
 gd:i = p4
 gm:i = (1+sqrt(5))/2
 twopi:i = 2*$M_PI

 lpf:k chnget "lpf"
 hpf:k chnget "hpf"
 fdb:k chnget "fdb"

 lpfs:k port lpf, 6
 hpfs:k port hpf, 6
 fdbs:k port fdb, 1

 sigin1:a = inch(1)*fdbs
 sigin2:a = inch(2)*fdbs

 del1:a delayr gm*gd
 fil1:a tone  del1, lpfs
 fil1:a atone fil1, hpfs
 del2:a delay sigin1 + fil1, gd
 fil2:a tone del2, lpfs
 fil2:a  atone fil2, hpfs
      delayw sigin2 + fil2

 ph:a phasor  1/(gm*gd)
 sigc:a = cos(twopi*ph)
 sigs:a = sin(twopi*ph)
 l1:a,r1:a pan2 del1,sigc*sigc,2
 l2:a,r2:a pan2 del2,sigs*sigs,2

 chnmix (l1+l2)*0.5, "left"
 chnmix (r1+r2)*0.5, "right"
endin

instr 2
 d1:i = 0.013
 d2:i = 0.011
 modmax:i = 0.02
 modamp:i = 0.0023

 l:a clip chnget:a("left"),2,0.75
 r:a clip chnget:a("right"),2,0.75

 mod:a = modamp + oscili(modamp, 0.95)
 del1:a flanger l,mod+d1,0,modmax
 del2:a flanger r,mod+d2,0,modmax
 l = del1*0.25 + del2*0.75 + l
 r = -(del1*0.75 + del2*0.25) + r
 l,r reverbsc l*0.5,r*0.5,0.7,5000
   out r, l

 chnclear "left"
 chnclear "right"
endin

</CsInstruments>
<CsScore>
```

```
i1 0 z 12
i2 0 z
</CsScore>
</CsoundSynthesizer>
```

While in this particular application only one instance of instrument 1 is employed, if multiple delay lines are required, the program can be scaled up by the use of several copies of this instrument. Moreover, the code can be expanded to use multiple channels for both inputs and outputs.

3 Musical Approach

The Csound program in itself does not do anything, it only provides an open canvas for a performer to express musical ideas. It is one of the components of the *Frippertronics* genre; what complements it is a sound source of an appropriate instrumental nature (e.g. the electric guitar). In my case, I chose a synthesizer with a comprehensive user experience implementation to provide a wide palette of sounds and support an expressive performance (also matching my own skills and training as a musician).

The musical work is fully improvised, there is no previous direction or structure beyond a few general guidelines, such as

- a two-part form was used.
- a set total duration (ca. 30 min) was agreed.
- each part was intentionally designed to have an asymmetric arch structure, with a culminating point at around 2/3 - 3/4 length.
- a modal approach to improvisation was chosen, which was dissolved and restored at different times along the piece.

This final element seems to me to be a key aspect of the genre, possibly related to the fact that modes can be brought to unify the various elements of sonic texture, bringing melodic/harmonic ideas close to timbral/spectral aspects of the musical flow.

4 Conclusions

This short paper reports on the performance of *Frippertronics* at the Klangtheater during the ICSC 2024. It discusses the motivations behind the musical work and describes the technical details of the electronics element of the piece. This is complemented by a brief commentary on the musical approach used in the performance.

This instance of *Frippertronics* has been released as an album [5]. I would like to thank the Musical Acoustics Department at dwn, for providing the opportunity, and for capturing the concert as a binaural recording.

References

1. Fripp, R. and Eno, B.: No Pussyfooting (1973) https://open.spotify.com/album/7090pUnNlv1lklI2lI2X6J?si=owrqKlHMTaCYXGYoL2F1dg
2. Lazzarini, V.: Frippertronics (2023) https://open.spotify.com/album/04gGsSIfWECdp0hP2ptIcS?si=gbGnD_aMRUClfPLW77JhUQ
3. Lazzarini, V. et al.: Csound: A Sound and Music Computing System. Springer (2016)
4. Csound Github site, http://csound.github.io
5. Lazzarini, V.: Frippertronics Live at the Klangtheater, dwn, Vienna (2024) https://open.spotify.com/album/6bfibWmuoAf3OlxGQbdMaz?si=Dzejr-fWSvSHdZreJEOdDQ

Living Csound

Steven Yi

stevenyi@gmail.com

Abstract. A meditation on Csound as living software and reflections on living with this program exploring sound and music. In this talk, I will look at Csound 7, the newest generation of our software, and discuss what it offers us today as users and as a community. I will discuss where we are today, as well as short- and long-term plans, and offer some thoughts on what we can do to nurture this program to keep it vibrant and healthy for the days ahead.

Keynote video recording: https://doi.org/10.21939/icsc2024-Yi

Why bother? The value(s) of an interface

Pierre-Alexandre Tremblay

Conservatorio della Svizzera italiana
tremblap@gmail.com

Abstract. As everyone attending this conference will know very well, creative coders have, today more than ever, a breadth of options to make music programmatically: from specialised software old and new, to toolset expanding general computer languages, many visions of what a good art-enabling coding environment cohabitate and cross-pollinate. While trends rise and fall, along the way communities wax and wane, the artworks survive as best as they could, and the artist-programmer tries to strike a balance between inspired mastery and catching up.

But is there a value to this multitude of opportunities? Are new proposals diluting energies and foci? Are there commonalities that would be better sorted once-and-for-all? What values each of these interfaces defend, consciously or not? And what about the underlying metaphors they employ to create bridges between practices and disciplines?

In this presentation, the author will muse on these questions around the design of software environments that are foundational to artistic research through creative coding. He will try to ascertain their value, the affordances and responsibilities of such enabling endeavour, through sharing his early-career personal experience of CSound, and the emergence of the FluCoMa ecosystem.

Keynote video recording: https://doi.org/10.21939/icsc2024-Tremblay

ROUNDTABLE SESSION

Roundtable – Future developments in Csound and its community

Joachim Heintz[1] and Alex Hofmann[2]

[1]HMTM Hannover, DE
[2]mdw – Univ. of Music and Perf. Arts Vienna, Dept. of Music Acoustics – Wiener Klangstil, Vienna, AT
[1]joachim.heintz@hmtm-hannover.de
[2]hofmann-alex@mdw.ac.at

Abstract. This roundtable serves as a platform where Csound Users and Csound Developers to have an in-depth exchange on the future development of the software. Planned topics include, but are not limited to, the overall development status of Csound, the Csound plugin systems and the documentation of Csound.

Keywords: Csound development, Core developers, Csound Users, Csound Community, Qualities, Issues.

1 Description

Csound has undergone a significant development over the last two decades [1, 3]. This applies to the extension of the coding language and a number of different usage cases such as on embedded platforms (e.g. Raspberry Pi, BELA [4]), but also applies to the structure of open source software development in general and the inherent community effort [2]. In this roundtable we motivate a discussion between Csound developers and Csound users on the following topics, and beyond:

Csound Development

- How do the developers see the current procedure of Csound development? What is good, what is missing?
- What could be desired contributions from the users?
- What tasks need to be addressed? Who can work on these tasks?

Csound Plugins

- What is the general status on this development?
- Why are there currently two plugin platforms (repositories) [5, 6]? Can they be unified?
- What is the workflow for users?
- Which jobs need to be done, and who can do these jobs?

Csound Documentation

- State of Csound Manual, FLOSS Manual, and other parts of the documentation.

- User feedback: What is good, what is missing?
- Which contributions are welcome by the maintainers?
- Which jobs need to be done, and who can do these jobs?

2 Biography/CV of Organiser(s)

Joachim Heintz uses Csound since 1996 to realize his compositions, installations, improvisations. He became more active in the Csound community around 2004. Since 2010 he is maintaining the Csound FLOSS Manual, and recently also the Csound website csound.com, together with an international group of Csounders. Together with Alex Hofmann he hosted the first ICSC in Hannover in 2011.

Alex Hofmann was a co-editor of the Csound FLOSS Manual's first edition and a co-organiser of the first ICSC in Hannover in 2011. Together with Bernt Isak Wærstad and Victor Lazzarini, he worked on porting Csound to embedded platforms such as RaspberryPi in 2013 and BELA in 2018.

3 Technical requirements

No special technical equipment is required.

3.1 Duration

1-2 hours.

Acknowledgements

This research was funded in part by the Austrian Science Fund (FWF) [10.55776/AR743].

References

1. Csound Github site, (http://csound.github.io/, accessed 31.7.2024)
2. Heintz, J., Hofmann, A., & McCurdy, I. (2011). Csound - Floss Manual (First Edition). Floss Manuals.
3. Lazzarini, V. et al.: Csound: A Sound and Music Computing System. Springer (2016)
4. Waerstad, I.B., Hofmann, A. (2019). Csound and Bela: touching opcodes, Online publication in the Bela-WebBlog (https://blog.bela.io/learn-csound-with-bela/, accessed 31.7.2024)
5. https://csound-plugins.github.io/csound-plugins/ (accessed 2.10.2024)
6. https://github.com/csound-plugins/risset-data (accessed 2.10.2024)

WORKSHOP SESSION

Developing Csound

Steven Yi

stevenyi@gmail.com

Abstract. This workshop introduces users to the tools, processes, and practices involved in building and developing Csound. Attendees will use popular IDEs and editors to build, explore, debug, and optimize the Csound codebase.

Keywords: Csound, Development, IDE, Build, Debug, Optimize, Cross-platform.

1 Description

This workshop introduces users to the tools, processes, and practices involved in building and developing Csound [1]. Attendees will go through a series of exercises using popular IDEs (Xcode [2], Visual Studio [3]) and editors (Visual Studio Code [4]) to build, explore, debug, and optimize the Csound codebase. The target audience is Csound users with intermediate programming experience who may be new to C/C++ de-velopment and are interested to customize Csound for their own use as well as make contributions for the benefit of the community.

Planned activities include:

- Building Csound: Understanding the build system, setting up your tools, and diagnosing issues with builds
- Tour of Csound codebase: overview of layout of codebase; walkthrough of key data structures; a guided tour of the parser, engine, opcodes, library functions, and I/O
- Debugging Csound: work through exercises using tools (unit tests, debuggers, audio editors for waveform exploration) to diagnose and fix bugs
- Optimizing Csound: working through exercises to diagnose performance issues with internal Csound code as well as Csound CSD projects using a profiler
- Beyond the Desktop: A brief discussion and walkthrough of Android, iOS, WebAssembly, and other platforms and builds
- Questions and Answers

2 Biography/CV of Organiser(s)

Steven Yi is a composer, performer, and music software developer. He is a core developer and maintainer of Csound; the author of Blue, a music composition environment for Csound; creator of the csound-live-code library for live coding with Csound; co-creator of the Csound Web-IDE; and author of various WebAudio Csound projects.

Steven is a long-time supporter of free and open source software for music. He has presented at the Interna-tional Computer Music Conference, Linux Audio Conference, and Csound Conferences. In 2016, Steven re-ceived his PhD from Maynooth University for his thesis work on "Extensible Computer Music Systems."

More information about Steven is available at his website: http://www.kunstmusik.com.

3 Technical requirements

Ideally, attendees will bring their own laptops for this workshop. For attendees without laptops, lab ma-chines are necessary that permit installing software tools and that have ample local disk space (to accommo-date the size of tools and build space required). Using laptops is preferred so that users will go through the exercises on their personal machines and can take their experiences home to continue on with their explo-rations. Internet access is a must for this workshop. Due to the proposed length of the workshop as well as general high power cost for compiling code and rendering with Csound for testing, easy access to power outlets will be necessary for attendees. A projector will be necessary for demonstration purposes. A lab space that affords easy movement for me to assist attendees as well as for attendees to work together would be preferred.

3.1 Duration

4 hours

- Building Csound: 1 hr (includes walk throughs setting up and building with command-line as well as with Xcode (macOS), Visual Studio (Windows), and Visual Studio Code (Linux))
- Tour of Csound codebase: 30 minutes
- Debugging Csound: 45 minutes
- Optimizing Csound: 45 minutes
- Beyond the Desktop: 30 minutes
- Questions and Answers: 30 minutes

4 References

1. Csound Github Project, http://github.com/csound/csound
2. Xcode, https://developer.apple.com/xcode/
3. Visual Studio, https://visualstudio.microsoft.com/
4. Visual Studio Code, https://code.visualstudio.com/

INSTALLATION SESSION

Web Box: Surveillance and Manipulation in the Digital Age Trans-interactive installation for physical and web environments

Lorenzo Ballerini[1], Giuseppe Ernandez[2] and Massimo Reina[3]

[1,2,3]Conservatory of Trapani, Italy
[1]lorenzo.ballerini@constp.it

Abstract. In our society, an illusory freedom conceals pervasive surveillance, with socioeconomic mechanisms monitoring our actions and subtly guiding our behavior. This control is exerted through advanced computer systems, especially the Web, which functions as a complex device integrating linguistic and nonlinguistic elements, regulations, and institutions to maintain capitalist power dynamics.

This installation challenges the digital control system by interweaving the real and virtual worlds. At the center of the exhibition is a glowing, resonant black box, a monolithic symbol of mystery and hidden knowledge. This monolith, an archetype of the digital deity, emanates its own light and sound by absorbing and interpreting data from a dedicated web page, accessible via a QR code, allowing visitors to interact with its virtual counterpart. In turn, the monolith reacts by altering the screens of smartphones connected to the webpage, highlighting the often invisible processes of digital surveillance and social manipulation.

Through this interaction, the installation reveals how simple actions generate information streams, highlighting the pervasive and opaque nature of digital control in contemporary society.

By exploring Csound and its Web engine, we want to offer a trans-interactive experience that evokes awe and unease, prompting reflection on the influence of the digital world on our real relationships. The monolith and its digital black box counterpart symbolize the hidden forces that shape our destinies, encouraging visitors to critically confront the pervasive surveillance of contemporary society.

Keywords: Society of control, digital surveillance, manipulative power, virtual interaction, Web interaction, trans-interactive experience.

1 Program notes

We live immersed in a society of control in which, through an illusory system of freedom, everything we do is tracked and can be somehow used, with or without our consent. Socioeconomic mechanisms constantly monitor us, infiltrating our everyday lives and exerting their power by making us do exactly what they want. The challenge is to discover new approaches to navigate this transition, avoiding both despair and illusion, but rather seeking innovative tools. [1].

The society of control operates through third type machines, digital systems, and computers, whose passive danger is blurring and the active one is hacking and the introduction of viruses. [1] The quintessential computer system today is the Web, understood as the entire complexity of the global telematic network, consisting of a vast collection of interconnected pages, and accessible through software specially designed for this purpose.

The Web is a device, and as such it is part of a complex network that unfolds in a diverse whole, involving conversations, institutions, physical structures, regulations, laws, administrative decisions

and scientific statements. This concept encompasses virtually everything, both linguistic and nonlinguistic, and constantly plays a concrete strategic role within power dynamics. [2]

Thus, the purpose of the device is to respond to an urgency and achieve an immediate effect. From a philosophical point of view, the Web can be considered a complex entity that raises several questions and reflections.

The point we want to highlight is the manipulative power of the web, driven by a capitalist system that shapes our behavior and choices. This installation, bridging real and virtual worlds, aims to challenge society's control system using its own tools.

The work invites the viewer to confront a glowing, resonant black box, capable of emanating its own sound and light and placed at the center of the room. A divine monolith, an archetype of mystery and hidden knowledge that absorbs information from a dedicated web page, accessible via a QR code, through which the visitor can interact with the virtual copy of the physical black box present in the space.

The monolith, like an enigmatic deity, consumes this information and interacts unpredictably, feeding on the visitor's every gesture. As visitors engage with the virtual black box, their dehumanised gestures are transformed into data that flows directly to the physical one. Usually hidden from us, these messages will appear on the visitor's web page screen, revealing how a simple action, like zooming in, can turn into a continuous stream of information, even of a broader spectrum such as data location, interaction time, quality, and more.

In a similar way, the interaction goes both ways, from the real monolith to the virtual page. The monolith sends information back to each visitor's web page, creating changes on the screen that the visitor cannot control, instilling new information and manipulating their perception.

By creating a trans-interactive, relational experience, this installation embodies the principles of Nicolas Bourriaud's "relational aesthetics," [3] focusing on the social interactions and shared political conditions that shape individuals.

This experience is designed to evoke a sense of awe and unease, revealing the hidden mechanisms and prompting reflections on how the digital world manipulates and transforms our real-life relationships. The monolith becomes a symbol of the opaque and inscrutable nature of digital surveillance, collecting and reflecting back the data we unknowingly provide, thus uncovering the unseen forces that influence our daily lives, much like a hidden god shaping our destinies from the shadows.

The main concept of interaction between the webpage and the physical Black Box is as follows: visitors will be able to interact with a virtual Black Box accessible through the webpage. Navigating the virtual environment, every gesture - such as zooming in/out, scrolling, and spatial movements - will generate data sent to the real counterpart of the Black Box. Simultaneously, this data will be displayed on the visitor's screen, showing in real-time the amount of information generated by each simple action.

In this way, the data that usually remains hidden from the user will be visible, creating a sort of "console log" directly on the webpage. Thus, what is typically accessible only to developers becomes transparent to the visitor as well, enabling a deeper understanding of the data flow each interaction entails.

We want to highlight the interaction between the visitor and the Black Box, explaining clearly how the data generated by user actions are displayed in real-time, making the flow of information normally hidden evident.

We aim to avoid an interaction where visitor gesture/sound/light is directly controlled, specifically steering clear of transforming the physical cube into a mere "synthesizer" for the sole purpose of playful interaction or interactive exercise. The concept is precisely to make visible and simultaneously

uncontrollable by the user the transmission of information, which will then be arbitrarily managed by the Black Box.

In turn, the Black Box can send messages to visitors (clients) to modify the webpage they are navigating. In the video we presented, the glitches on the webpage are linked to the percussive sounds of the physical Black Box, which sends real-time "disturbance" messages online. This further emphasizes a sense of powerlessness in the face of the physical entity that can manipulate both the real and virtual environments.

JavaScript, WebSocket, Node.js, Three.js, Socket.io, Csound, and Max MSP are all essential components for the technical implementation of the installation. These APIs and tools collaborate to manage data flows, interactions, sound, and light. JavaScript provides the foundation for interactive front-end programming, Node.js server functions, WebSocket and Socket.io facilitate real-time communication between client and server, and Three.js enables the creation of 3D graphics.

A notable element is Csound, which not only generates and manipulates sound but can also integrate with the web. Csound can communicate with JavaScript and other APIs, allowing direct interaction between sound generation and web events. This enables the creation of an interactive and dynamic audio experience that responds in real-time to user actions.

Max MSP is used for further audio processing and light control, completing the overall integration of the various elements. Together, these components make possible an interactive and immersive experience for users, synergistically managing data, sounds, and lights.

The integration of Csound into web-based applications signifies a pivotal advancement in digital sound design and interactive user experiences. This fusion of Csound's powerful audio synthesis capabilities with the accessibility and interactivity of web technologies opens up a myriad of possibilities for creative expression and sonic exploration. Through rigorous experimentation and research, we are delving into this exciting frontier, paving the way for innovative developments at the intersection of music, technology, and the web. The paper presented for ICSC 2024, "The Internet of Sound," [4] is also related to this context.

2 Biography/CV of Composer, Creator and Performers involved

Current position, Professor of Computer Music at the Conservatory of Trapani, Italy.

The work presented is in collaboration with my students Giuseppe Ernandez and Massimo Reina, who have excelled in their engagement, technical, research and artistic skills.

Lorenzo Ballerini is composer, sound and new media artist, mainly focused in live electronics and multimedia installations. Born in Florence in 1990. Graduated in Music and New Technologies at Luigi Cherubini Conservatory in Florence and in second-level master's degree AReMus (Artistich Research in Music) at Santa Cecilia Conservatory in Rome.

His aesthetics revolves around exploring the political and social implications of human connection to art within today's society.

He has participated as composer and performer in festivals such as ADE Festival, Artech2023, Berlin Biennale, Bright Festival, CHB Berlin, Diffrazioni Festival, Fabbrica Europa, Gaida Festival, MEFF, SMC2018, SMC2019, Tempo Reale Festival.

He has collaborated with artists including Alvise Vidolin, Christine Meisner, Michele Marasco, Nicola Sani, Paolo Parisi, Roberto Fabbriciani, Tiziano Manca. 2022

In 2022, Ballerini assumed the role of Professor of Electroacoustic Music at the Conservatory of Trapani, Italy, where he shares his expertise with aspiring musicians and composers. In 2023, he broadened his horizons through an internship in the Sound Art Department at UDK, University of the Arts, Berlin, while concurrently serving as a Professor of Computer Music at the Conservatory of Pavia, Italy.

3 Technical requirements

The physical installation consists of a white column at the center of the room, upon which a black cube will rest. This structure will be custom-built for the event, and we can determine its dimensions together with the committee to suit the event's needs and the room's size.

Approximate dimensions:
- Column: 50 x 50 cm, height 80 cm
- Black Box: 50 x 50 cm, height 50 cm
- Materials: wood and aluminum.

Figure 1 - Installation, The Black Cube

Inside the black cube, there will be actuators (exciters) and RGB LEDs for the propagation of sound and light, all controlled by a PC located within the stand. No additional speakers are necessary. Two small spotlights will be needed to softly illuminate the environment for the visitor. Ideally, the setting should be an indoor, isolated space, as free from external light as possible.

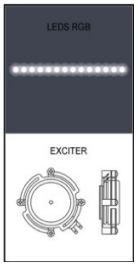


Figure 2 - Vibrant Transducer and LEDs inside the object

The webpage will be accessible via a QR code printed next to the work's description. It will be specially designed for both iOS and Android smartphones and tested on all browsers. The webpage will be online, allowing visitors to connect through any network, such as 5G or Wi-Fi. There is no need for a local network connection. This approach ensures the most user-friendly and immediate navigation experience, without any additional steps.

Figure 3 - Web Page

Technical Rider

What we have:

- Physical Installation with exciters and LEDs
- Approximate dimensions: column 50 x 50 h 80 cm, Black Box 50 x 50 h 50 cm
- PC for managing the entire system

What we need:

- Power extensions (to supply electricity from the outlet to the center of the installation)
- Power strips (2 x 6 outlets)
- Wi-Fi or wired network connection
- 2 x movable spotlights for soft lighting
- No additional speakers or lights are required besides the two spotlights
- We will need a shipping address to send (both ways) the package containing the installation.

Setup and testing time: 3-4 hours.

3.1 Duration

The installation can run indefinitely without supervision, opening times and/or personnel are not required.

3.2 Supporting materials

Video - Preview Web App for visitors (Audio will emanate from the physical Black Box):
https://drive.google.com/drive/folders/1uNQ72QSRH9oobUF_yfdY3FkoMzPBPu0-?usp=sharing

Pictures:
https://drive.google.com/drive/folders/1XsDVLCDrYEqM9TOU38UAMnOXIBh7iBai?usp=sharing
Web App Reverb with Csound, related to the paper presented for ICSC 2024, "The Internet of Sound:" https://github.com/csytp/WebCsoundVerb

Other Works related to the topic of the presented work
Transimmanency – Transinteractive Installation for Web and Bright Resonant Objects, 2023:
https://www.lorenzoballerini.info/works/#transimmanency
Transimmanency - Related Paper Published for ARTECH2023:
https://dl.acm.org/doi/10.1145/3632776.3632814
Circular - Participatory Installation, 2023:
https://www.lorenzoballerini.info/works/#circular
Aeterna – Live Elctronics and LEDs Visual, 2023:
https://www.lorenzoballerini.info/works/#aeterna
Digital Relations – Live Elctronics and LEDs Visual, 2018:
https://www.lorenzoballerini.info/works/#digital_relations

References

1. Deleuze, G.: Postscript on the Societies of Control. L'Autre Journal 1, (1990). Included in the forthcoming translation of Pourparlers. Editions Minuit, Paris. To be published by Columbia University Press
2. Foucault, M.: Dits et Ecrits, tome 1. French & European Publications, (1954-1975)
3. Bourriaud, N.: Relational Aesthetics. Trans. S. Pleasance & F. Woods. Les Presses du Réel, Dijon (2002)
5. The Internet of Sound paper, Web Verb example, https://github.com/csytp/WebCsoundVerb

Polyomino Interface for Pitch Lattices

Tim-Tarek Grund

mdw – Univ. of Music and Perf. Arts Vienna, Dept. of Music Acoustics – Wiener Klangstil, Vienna, AT
grund@mdw.ac.at

Abstract. This sound installation is using Csound to explore pitch lattices. There are several online applications that allow users to explore pitch lattices. However, few tangible interfaces for this purpose exist. The polyomino interface for pitch lattices aims to bridge this gap by providing a grid of fiducial markers representing pitches that can be played by covering them with geometric shapes (polyominoes). Moving, rotating and exchanging these pieces allows users to explore pitch relations in an intuitive way.

Keywords: Pitch lattice, just intonation, tetromino, polyomino

1 Program notes

The presented sound installation is based on the principles of pitch lattices. Organizing musical pitches in geometric shapes is fundamental to the design of musical instruments and has major implications for musical performance. However, not only musical performers benefit from an effective pitch layout: Composers and mathematicians have been experimenting with pitch lattices as early as 1739 [1]. Nowadays, several online applications exist, that allow users to test out pitch representations and combinations. One such representation is the 2,3,5 square pitch lattice. It organizes pitches in major thirds with a frequency ratio of 5:4 from left to right and in perfect fifths with a ratio of 3:2 from bottom to top [2]. This sound installation uses a grid of 11x7 AprilTag [3] fiducial markers. Each marker encodes the number of an individual field of the pitch lattice. A camera films this grid from above and a python script detects visible markers. Using a set of polyomino elements users can cover fields of the grid. The python script then registers covered fields. After pressing an updating button, the audience can send Open Sound Control (OSC) information of the field number and corresponding pitch to a Csound program, which starts to play tones with the pitches of the covered fields. One slider changes the volume, another alters the global tuning between 5-limit tuning and 12 tone equal temperament tuning (12TET). For more information, please see the *Polyomino Interface Explanation and Demo* in Section 3.1 Supporting Material. Polyominos are shapes consisting of n unit squares that share at least one edge with another unit square, the most famous ones being tetrominoes (four unit squares, used in the game *Tetris*). Placing tangible shapes on a lattice allows users to quickly test out pitch combinations and tuning systems. Furthermore, translation, rotation and reflection of these shapes can serve as inspiration for writing music. Giving these shapes a physical form invites users to engage with it in a tangible, embodied manner. The interface makes extensive use of Csound's efficient voice allocation system. Csound receives the number of the fiducial marker and the frequencies in 5-limit tuning and 12TET. One Csound instrument handles OSC information and voice allocation, another synthesizes the sound. The OSC handler then creates a unique note name for the field, consisting of the synthesis instrument's number and a fractional note name. This allows for turning individual notes on and off. Csound also receives volume and pitch morph values. These are handled as global k-rate variables in order to affect all currently playing notes. The use of tetrominoes in a 2,3,5 pitch lattice has already been explored by [4].

2 Biography/CV of Composer, Creator and Performers involved

Tim-Tarek Grund is a Berlin-born, Vienna-based music technologist. One of his fields of interest is computer vision. He has worked with anatomical landmark detection in a granular synthesis audio installation called *Cloud hands* (see Section 3.1 Supporting materials). He is currently researching live-electronic setups and mapping strategies.

3 Technical requirements

The setup for the installation can be seen in Section 3.1 Supporting materials under *Setup* . I will provide the AprilTag plate, polyominoes, the updating element, a computer, an audio interface and the camera. The setup requires a set of headphones, a table and a monitor screen. The installation needs to be indoors. It would be preferable to hide the computer. I can also be present to supervise the installation and give instructions to the audience. The installation can be run indefinitely. I would like to provide a demonstration once or twice a day and supervise the installation and give instructions to the audience, if part of the equipment cannot be locked away during open hours.

3.1 Supporting materials

Polyomino Interface Explanation and Demo - https://youtu.be/BxDidLPdXok
Polyomino (Github repository) - https://github.com/grundton/polyomino
Setup – https://github.com/grundton/polyomino/blob/main/img/Setup.png
Cloud hands (Github repository) - https://github.com/grundton/cloud_hands

4 Acknowledgements

This research was funded in whole by the Austrian Science Fund (FWF) [10.55776/AR743]. For open access purposes, the author has applied a CC BY public copyright license to any author accepted manuscript version arising from this submission. I would like to thank Alexander Mayer for his guidance and support with 3D printing the physical interface.

References

[1] Euler, L. (1739). *Tentamen novae theoriae musicae*.
[2] Fonville, J. (1991). Ben Johnston's Extended Just Intonation: A Guide for Interpreters. *Perspectives of New Music*, *29*(2), 106–137. https://doi.org/10.2307/833435
[3] Olson, E. (2011, May). AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation* (pp. 3400-3407). IEEE.
[4] *mannfishh* [YouTube channel] (2024, April 25) *MICROTONAL TETRIS* [Video]. YouTube. URL https://www.youtube.com/watch?v=CSL_Axohw94

Csound-FPGA Integration

Aman Jagwani[1] and Victor Lazzarini[2] *

[1,2]Department of Music, Maynooth University
[1]aman.jagwani.2023@mumail.ie
[2]victor.lazzarini@mu.ie

Abstract. With the development of Bare-metal Csound, embedded systems with ARM-based CPUs can now be targeted to run Csound audio programs. This installation will demonstrate the potential of this development through an interactive, generative Csound piece running on a Digilent Zybo Z7020 board, which contains a Xilinx Zynq 7000 SoC. Csound's generative and synthesis capabilities will be interfaced with motion-sensing through LIDAR sensors to capture and convert motion in any of the common spaces of the conference into varied ambient sonic results. The purpose of this installation is to create an interactive ambience for a common space and to showcase the potential and portability of Bare-metal Csound.

Keywords: Bare-metal, FPGA, Interactive

1 Program Notes

Field Programmable Gate Arrays(FPGAs) are integrated circuits comprising of a huge numbers of configurable logic units. They provide the benefits of ultra-low latency, high sampling rate, high throughput and reprogrammable hardware, in the context of audio applications. Commonly, FPGAs are packaged as a system-on-chip(SoC) with an on-board processing system or CPU with ARM architecture [3]. Now, Csound 7 can be cross-compiled to target various bare-metal (no-OS) chips or boards that have ARM-based CPUs, including the CPU on an FPGA SoC. This opens up several opportunities to integrate with and leverage the power of FPGAs. This installation will demonstrate this potential.

This installation entails a generative, interactive Csound piece running stand-alone on a Digilent Zybo Z7020 board [6] that includes a Xilinx Zynq 7000 FPGA SoC[1]. Bare-metal Csound runs on the ARM Processor of the Zynq, generating and sending synthesized audio to the FPGA part of the chip where it is processed by accelerated audio processing FPGA modules. These modules are ports of complex Csound opcodes such as reverbsc that will be running sample-by-sample, with ultra-low latency. Additionally, Csound's generative capabilities such as randomization and sequencing are also used to control sound synthesis running on the FPGA in tandem, showcasing two different ways in which this platform can be leveraged.

The interactive portion of the installation is created using LIDAR [5] sensors, which will enable motion-based response for the generative Csound piece, modulating synthesis parameters, event scheduling and stereo spatialisation.

The entire installation runs on the Zynq 7000 chip, sending audio out of the audio codec on the Zybo Z7020 board.

2 Biography/CV of Composer, Creator and Performers involved

Aman Jagwani, from Mumbai, India, is a musician, music technologist, and researcher. He completed his Bachelor's degree in Electronic Production and Design and Performance at Berklee College of Music, Boston, and later earned a Master of Science in Sound and Music Computing from Maynooth University. Currently, Aman is undertaking his Ph.D. at Maynooth University under the supervision of Prof. Victor Lazzarini. His research centers on embedded systems, FPGAs, interactive systems and digital signal processing. Aman also has experience with interactive and immersive sound and art

* Aman Jagwani would like to thank the Hume Scholarship Scheme from Maynooth University

installations, having contributed to projects in India, Europe, and USA. In addition to his academic pursuits, Aman is involved in the Jazz and Electronic music scenes as a composer and performer. He often integrates custom Csound plugins and instruments into his performance work, merging his technological insights with his musical ability.

Victor Lazzarini is a Professor of Music at Maynooth University. A BMus graduate of the Universidade Estadual de Campinas (UNICAMP) in Brazil, he received his AMusD from the University of Nottingham, UK. His academic interests include musical signal processing and sound synthesis; computer music languages; electroacoustic and instrumental composition, and improvisation. He has authored over one hundred fifty peer-reviewed publications in his various specialist research areas. He is the author of Aulib and Aurora, two object-oriented libraries for audio signal processing, and is one of the project leaders for the Csound sound and music programming system. Recent papers include 'Parallel Computation of Time-Varying Convolution' (Journal of New Music Research 2020), 'Improving the Chamberlin Digital State Variable Filter' (Journal of the Audio Engineering Society 2022), 'Linear and Nonlinear Digital Filters: From the Analog and Beyond' (Computer Music Journal 2022), 'Issues of ubiquitous music archaeology: shared knowledge, simulation, terseness, and ambiguity in early computer music' (Front.Sig.Proc. 2023) and 'A Digital Model for the Prologue Voltage Control Filter' (Journal of the Audio Engineering Society 2024). He has also completed the volume Ubiquitous Music Ecologies (with D. Keller, N. Otero, and L. Turchet, Routledge 2020), as well a research monograph for Oxford University Press, Spectral Music Design: A Computational Approach (2021).

3 Technical Requirements

The only technical requirements for this installation will be a table, a stereo audio output and stereo speakers. This installation will be portable because it will only contain a Zybo Z7020 board and some LIDAR sensors. The location for this installation is flexible (provided no rain outdoors). It can also be moved between different spaces.

3.1 Duration

This installation can run indefinitely without supervision.

3.2 Supporting Materials

This video demonstrates a preliminary version of the piece and the sonic pallette, running on the same chip and board that will be used in the installation:

`https://youtu.be/sEWzhKcb3wc`

These videos demonstrate some of the author's previous installation work. The first installation's sonic aspect was entirely done with Csound:

`https://youtu.be/erDdpdK_O5U`
`https://youtu.be/EvB-BPgV3ww`

References

1. Xilinx Zynq-7000 SoC. Available: `https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html`. Accessed: April 28, 2024.
2. ARM Architecture. Available: `https://www.arm.com/architecture/cpu`. Accessed: April 28, 2024.
3. Jagwani, A.P.: Developing a Modular Sound Synthesis Platform for FPGAs with High Level Synthesis Techniques. Master's thesis, Maynooth University, Department of Music, 2023. Supervisor: Prof. Victor Lazzarini.
4. Kastner, R., Matai, J., Neuendorffer, S.: Parallel Programming for FPGAs. ArXiv e-prints, May 2018. Available: `https://arxiv.org/abs/1805.03648`.
5. Adafruit Microcontrollers. Available: `https://www.adafruit.com/category/689`. Accessed: April 28, 2024.
6. Digilent Zybo Z7: Start. Available: `https://digilent.com/reference/programmable-logic/zybo-z7/start`. Accessed: April 28, 2024.

Csound in the MetaVerse: CsoundUnity at Berklee

Richard Boulanger[1], Hung Vo (Strong Bear)[2], Xiaomeng Zhong[3], Ken Kobayashi[4] and Mateo Larrea[5]

[1,2,3,4,5]Berklee College of Music
[1]rboulanger@berklee.edu
[2]sbear@berklee.edu
[3]xzhong@berklee.edu
[4]kkobayashi4@berklee.edu
[5]mlarrea@berklee.edu

Abstract. This installation will showcase seven projects created and programmed in CsoundUnity by Professor Richard Boulanger's Electronic Production and Design students at the Berklee College of Music in Boston. Individual players and small groups will be able to choose from and enter immersive VR, AR, and XR worlds where they can: 1. Wander through Zhong's beautiful generative Sound Garden (*La forét*); 2. Design and play expressive Csound instruments in Kobayashi's Sound Lab (*Laser Synth*); 3. Jam together in mixed reality playing Cabbage and Csound-for-Live instruments in Vo's version of Ableton Live (*Effortless*); 4. Explore Larrea's Musical Island where players pick, toss, and plant "soundFruit," that grow on trees, and "soundShrubs," that sprout from the ground (*Trapped on the Island*); 5. Join a trio and perform Zhong's VR version of Boulanger's classic wiiMote piece (*CsoundQuest*); 6. Explore Larrea's latest real-time sampling and spatial DSP world where you make sounds in actual space, capture them in virtual space, and with Csound, you process them, and spatially sequence them (*XR Audio in XR*); or 7. Colocate to see and collaborate with multiple local and remote players as you and they create, hit, stretch, squeeze, contort, reshape, grab, pass, catch and launch Vo's "SoundOrbs" and "SoundWanders," under the stars, on the beach, over the rooftops, and under the sea (*Collaging in the MetaVerse*). These CsoundUnity worlds will be installed on a number of Meta Quest 2+3 MR Headsets and screencast onto multiple laptops. This will allow many to explore and play simultaneously while others can watch them play as they wait for an available headset to immerse themselves in these incredible VR soundworlds.

Keywords: Unity, CsoundUnity, VR, AR, XR, Quest, Ableton Live, Cabbage, Csound-for-Live, ZeroTier, immersive, multiplayer, colocation, mixed media

1 Program notes

Collaging in the MetaVerse: A local and web-based multiplayer environment for sound design, composition, improvisation and structured-scored performance. Hundreds of Csound instruments, textures, and timbres serve as starting points for interactive transformation, spatialization, mutation, and localization - malleable sound shapes correspond to morphing and mutating timbres which can be place in specific locations or tossed about to wander freely. Local and networked players can see each other, collaborate with each other on sound objects they are designing, toss objects to each other and steal objects from each other. Sounds can be further customized and controller gestures scaled and assigned from with the virtual space.

Effortless: Multiplayer jamming in Mixed-Reality with prebuilt Ableton Live sessions. Trigger Live clips, map Live parameters to controllers, use Live processing, Playing Live melodies, chords and loops, all from within the Quest. Moreover, one can play with live players (and see them playing) in the room via the Quest Passthrough mode. Players can play along with tracks prepared by Berklee EPD students. Or jam on their own creations. All will be playing built-in CsoundUnity instruments from within *Effortless* together with instruments running on the MacBook using and Csound-for-Live instruments in Max-for-Live and Cabbage AU and VST instruments and effects. The session, simultaneously running in VR and on the MacBook is synced to the session tempo via Ableton Link.

La forêt is an interactive musical garden, created for the purpose of escaping from the city and finding inner peace through sound. It is created using csoundUnity, and features generative music, and an interactable environment that allows players to customize their audio experience.

CsoundMetaQuest an immersive, multiplayer VR revision and reinterpretation of the Boulanger classic - *wiiSoundQuest,* premiered with wiiMote controllers at the very first ICSC in Hanover, Germany.

Laser Synth is a VR Csound instrument created in the Unity game engine. The musician grabs two orbs through which the Csound instrument is played. Distances control and filters cutoff. Other gestures control multiple parameters. The ranges of all parameters can be set on the screen within the environment and effects are also assignable, scaleable and controllable. One can also monitor the tuning of this expressive and customizable instrument while playing it.

Trapped on the Island was premiered at the 2022 ICSC in Shannon, Ireland. At that time, the system only supported a single player. Then, a quartet was presented with four players exploring the island on their own with no idea what sounds others were making or fruits other were picking or shrubs they were planting. For this installation, the interface has been redesigned, and the system has been expanded to support multiple local and network players, all on the same island, and all seeing and hearing what each other is doing making for a more musical and fun experience. All the fruits and shrubs are grown from the seeds of "Trapped in Convert" and additional sample-based and sample processing 'weeds' have also found their way into this 'synthetic' garden.

XR Audio in XR: The current paradigm of audio implementation in Extended Reality (XR) involves designing, positioning, and sequencing sound objects on a 2D computer screen before experiencing them in a 3D environment. However, advancements in computing power and XR development tools now allow for creating these experiences within the same medium, maintaining immersion throughout the process. XR Audio in XR proposes a mixed-reality experience where audio material is recorded through the Oculus Quest 3 microphone, edited using the digital signal processing capabilities of CsoundUnity, and sequenced using Unity's physics engine. The result is a 3D audio-

centered experience that enables users to sound design the ambiance of any physical space and fully utilize the mixing possibilities that spatial audio offers.

2 Biographies

Dr. Richard Boulanger is a Professor of Electronic Production and Design at the Berklee College of Music in Boston where he has been working with some of the most talented and creative sound designers, songwriters, performers, composers and innovators for the past 38 years.

Hung Vo, (Strong Bear), from Vietnam, recently graduated from Berklee College of Music, Electronic Production and Design major. He holds a B.E. degree in Electronics and Telecommunications from Posts and Telecommunications Institute of Technology in Ho Chi Minh City, Vietnam and a M.S. Degree in Computer Science from Clemson University. He is the founder and CEO of *Designveloper*, a software design and development company, since 2013.

Xiaomeng (Susan) Zhong is a sound designer and audio engineer with a passion for creating immersive experiences in games and multimedia, with a Bachelor in Electronic Production and Design from Berklee College of Music. Her experience includes designing custom sound effects, remixing, composition, foley, and creating interactive performative audio systems in game environments. Susan will be starting her Masters in Media Arts and Technology at UC Santa Barbara in September.

Ken Kobayashi was born in Texas to a Japanese-American family and started playing violin at the age of four. Now in Boston, they graduated from Berklee College of Music and are seeking a Master degree in computer music, blending their love for music and programming. Their interests span vocal synthesis, programming, and Japanese and Korean pop.

Mateo Larrea explores the essence of perception, learning, and creativity by crafting rich, multimodal immersive experiences. Currently, he is a Virtual Reality / Feel Engineer at the ed-tech startup *Prisms*, specializing in creating and implementing intuitive learning interfaces that engage sound, touch, and sight in virtual reality. In the fall, Mateo will be joining Stanford University's Center for Computer Research in Music and Acoustics as a graduate student. His research interests include procedural media, experiential learning, simulation, interface design, programming language design, brain-computer interfaces, psychoacoustics, and computer music.
Mateo completed his undergraduate studies at Berklee College of Music. His past experiences include serving as a Teaching Assistant for MIT's course 2.S972: Making Music in the Metaverse, conducting research as an undergraduate at the Berklee Psychology of Music Lab, and working as an undergraduate researcher at Boulanger Labs on the CsoundUnity project.

3 Technical requirements

1. Four small tables (to hold the laptops). One longer table to hold the Quest2 and Quest3 VR Headsets. One MicStand with Boom. One MIDI keyboard controller stand. This could be set up anywhere indoors (best if it is in a location where attendees congregate so that they could just try it when they get a free minute – possibly in a public place near the coffee). We do need some space (to stand around and walk around and for multiple users to move arms freely). Essentially, four 'hosts' will assist and guide users as they journey into the various Immersive CsoundWorlds of their choice. Sound can be heard softly from MacBook speakers or just in the Quest headsets. Two small Bluetooth speakers will be set up for *Effortless*. (They can be turned on or off as deemed appropriate.)

3.1 Duration

One could play in these soundworlds for short periods or get friends and colleagues together for longer jam sessions in *XR Audio in XR*, *Effortless*, *CsoundQuest*, and especially when *Collaging in the MetaVerse*. In some ways, this is a popup installation. Our team will have Quest2 and Quest3 headsets in their backpacks and could support individual or group play anywhere and anytime. We would hope to run it where there is a lot of flow traffic and whenever there are breaks before or after concerts and before or after workshops, in the mornings before sessions begin, and during coffee, lunch, and dinner breaks.

3.2 Supporting materials

Please provide links to supporting materials such as previous works, images, plans and additional technical information. Please make sure the links are permanent and accessible. Hence, use permanent links such as Google Drive, Dropbox, Youtube, or Vimeo, not links that will expire such as WeTransfer.

References

1. Csound site, https://csound.com/
2. CsoundQt site, https://github.com/CsoundQt/CsoundQt
3. Csound Manual site, https://csound.com/manual.html
4. Csound FLOSS Manual site, https://flossmanual.csound.com/
5. Cabbage Audio site, https://cabbageaudio.com/
6. Cabbage for Games site, https://forum.cabbageaudio.com/c/csound-for-games/10
7. Unity Download Archive site, https://unity.com/releases/editor/archive
8. MetaQuest3 site, https://www.meta.com/quest/quest-3/
9. Meta Developer site, https://developers.facebook.com/

FERNNAH Reading

Joachim Heintz

HMTM Hannover
joachim.heintz@hmtm-hannover.de

Abstract. The proposed event is more a performance than an installation. But with an installation it shares that visitors can come in and leave, can move closer or stay far, and take their own time. It can happen between other events in a corridor or corner, as we had it in Montevideo on the ICSC 2015. It should be noted that the text is in German; but the proposed event is not about "understanding" the text rather than experiencing the musical space.

Keywords: Performative installation, Reading, Text and Music

1 Program notes

The text FERNNAH (far-close) was written in 2022 as reaction to the *Mirror of Simple Souls* by mediaeval mystic Marguerite Porete. It consists of fragments from Porete's book, combined with fragments from now. (What is this *now*?) In 2024 I developed electronic sounds as accompaniment to reading three sections of the text.

The music is generated in real-time in Csound, in what I call an organic generative structure, meaning that the sound characters have interaction with each other, thus creating the form and single events.

2 Biography/CV of Composer, Creator and Performers involved

After studying literature and art history, Joachim Heintz began his composition studies in 1995 with Younghi Pagh-Paan and Guenter Steinke at the Hochschule fuer Kuenste, Bremen. He composes for instruments and electronics, concerts, installations and performances. With his software instrument ALMA he has improvised with many musicians around the world. He is a member of the Theater der Versammlung Bremen and writes texts which are published in journals and in his Schrenz Verlag. In the realm of software development, he is active in the open source projects Csound and CsoundQt. He is the head of FMSBW, the electronic studio in the institute for contemporary music at HMTM Hannover, and of the electronic department of Yarava Music Group in Tehran. As board member of the Hanover Society for Contemporary Music (HGNM), he organises and hosts workshops, discussions and concerts as encounters between traditional Asian instruments and contemporary music.

3 Technical notes

This sounds are generated in Csound in real time, and run by my own laptop or smartphone. Only one loud speaker is required (in the size of Genelec 8040). And one chair for me. A corridor or corner in the conference space would fit; any time between two events is possible (20 minutes minimum).

Joachim Heintz

3.1 Duration

20 minutes minimum, one hour maximum.

3.2 Link

https://joachimheintz.net/fernnah.html

PROGRAM NOTES

ATT...

Joachim Heintz

HMTM Hannover
joachim.heintz@hmtm-hannover.de

Abstract. Although music is sometimes considered to be abstract, this is not true. So I cannot imagine a way to write an abstract about the ATT... reality ...

Keywords: Music, Very, Concrete

1 Program notes

A minimalist study of the motion of an acceleration / desire / grasp -> deceleration / withdrawal / leaving -> staying / steadying / lasting, and an "accompaniment" through distant, intangible chords in quirky motion. A small salute to my teacher Younghi Pagh-Paan on the occasion of her retirement from teaching in 2011.

2 Biography/CV of Composer, Creator and Performers involved

After studying literature and art history, Joachim Heintz began his composition studies in 1995 with Younghi Pagh-Paan and Guenter Steinke at the Hochschule fuer Kuenste, Bremen. He composes for instruments and electronics, concerts, installations and performances. With his software instrument ALMA he has improvised with many musicians around the world. He is a member of the Theater der Versammlung Bremen and writes texts which are published in journals and in his Schrenz Verlag. In the realm of software development, he is active in the open source projects Csound and CsoundQt. He is the head of FMSBW, the electronic studio in the institute for contemporary music at HMTM Hannover, and of the electronic department of Yarava Music Group in Tehran. As board member of the Hanover Society for Contemporary Music (HGNM), he organises and hosts workshops, discussions and concerts as encounters between traditional Asian instruments and contemporary music.

3 Technical notes

This piece is completely written in Csound. Except very short recordings of voice all sounds are synthesized.
The code is online under: https://joachimheintz.de/stuecke/part/ATT_Partitur_Csound.pdf

3.1 Duration

1'49'' (one minute fourty-nine seconds)

Joachim Heintz

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

This is stereo only. If selected, I am interested in distributing in real time.

3.4 Link

https://joachimheintz.de/stuecke/musik/ATT.flac

Silence(d)

Marijana Janevska

marijana.janevska90@gmail.com

Abstract. The paper discusses the piece "Silence(d)" (2020) for voice and electronics by the composer Marijana Janevska. It includes a programm note text about the piece, a biography of the composer-performer, Technical rider, information about the duration and a link to a realization of the piece.

Keywords: Composition, Performance, Csound, CsoundQt

1 Program note

"Silence(d)" (2020) is a piece for female voice and electronics. The idea and inspiration about this piece came from a project, where I had a task to write a 30 second piece for solo voice concerning silence and immediately a question came to my mind: How does the silence of the silenced voice sound? This silence is not relaxing, but very loud.

2 Biography/CV of Composer, Creator and Performers involved

Marijana Janevska (1990, Skopje, Macedonia) is a composer, violinist and singer. She graduated violin performance and composition at the Faculty of music in Skopje. Since 2018 she lives in Hannover, Germany, where she finished her Master studies in compostion under the mentorship of Ming Tsao, Gordon Williamson and Joachim Heintz in 2020 at the Hochschule für Musik Theater und Medien Hannover. Her works have a lot to do with exploring various uses of text and voice to produce the musical material and the incorporation of physical movement into the musical gesture. Her pieces have been performed on numerus concerts and festivals among which:
Musik21 Festival, ZKM Next Generation, Klangbrücken Festival, TRAIECT Festival, Schallfront Reverie(s) Festival, Hell Wach Festival, Music BIENNALE, Tage für neue Musik in Zürich, TIEM Festival in Teheran, SONEMUS Festival, KotorART, International CSOUND Conference 2019 and 2022. She has collaborated with numerous ensembles: Quartet Mivos, PRE-ART, Ensemble Ascolta, Ensemble Adapter, Ensemble Mosaik, Comet Trio, Concerto Ispirato and so on.
She has won notable prizes and scholarships, among which: Kompositionspreis für Zeitgenössische Musik 2024 der Stadt Oldenburg, First prize at the "9- th Pre-Art competition for young composers" in Zurich, "Klaus Hubert Kompositionspreis" for electronic music in Hannover, Stipendien für Komponisten vom Niedersächsischen Ministerium für Wissenschaft und Kultur, Nothilfe/ Neustart Stipendium der Ernst von Siemens Musikstiftung, DMR Stipendien from Deutscher Musikrat.

3 Technical notes

The electronics in the piece is simple and consists of beats that are partly produced in Csound and

partly in a DAW program. The performance of the piece is done in CsounQt where the performer with a small Bluetooth device activates the groups of beats.

The piece needs the following technical requirements:
- Headset microphone
- 2 Speakers- one for amplificatrion of the voice and one for the sound of the electronics
- Lap-top with Csound and CsoundQt installed that will be on stage because the Bluetooth device has a limited distance range
- Interface with mínimum 1 input and 2 ouputs
- Bluetooth device (R400 Laser Presentation Remote is what I use)

I will bring my own Bluetooth device, Inteface and Lap-top with the Csound code prepared for performance.

3.1 Duration

The duration of the piece is approximately 6'40".

3.2 Category

The piece is for voice and electronics and belongs to the category:
Mixed pieces for instruments(s) and electronics

3.3 Channels

For the realisation of the piece 1 Input and 2 Output Channels are neccessary.

3.4 Link

This is a Youtube link from a realisation of the piece:
https://www.youtube.com/watch?v=GbFB5Qu3qj4

Solar

Leon Speicher

leon.speicher@yahoo.de

Abstract. "Space, the final frontier.." Since my youth I was fascinated with the imaginative influence the stars have on our culture and society. All the planets of our solar system have an influence on each other and as soon as a heavy enough object enters their gravitational field, they change their behavior and pathway. Similar things happen between us humans. We enter each others life, have an influence and then we leave (or get kicked out).

Keywords: Computermusic, Csound, fixed media

1 Biography/CV of Composer, Creator and Performers involved

Leon Speicher, is a composer and electrical guitarist, who studies composition in the 4th semester at the HMTMH.
He was born 1997 in Hildesheim and started out1aying mostly Jazz, Rock and Blues music. Through experimentation and composing in- and outside these genres, he developed an interest in composing 'new music'.

2 Technical notes

The piece was realized entirely with Csound. It uses a mix of additive and FM synthesis. I programmed in a form of action/reaction that morphs the spectrum of the main voices. This will also apply to the spatialisation in the multichannel setup.

2.1 Duration

5 Minutes

2.2 Category

Electroacoustic music on fixed medium

2.3 Channels

Number of channels is 21.2

2.4 Link

https://drive.google.com/drive/folders/1vsusL7QEiJjnqe_kk6MfvvKcuT2nZ-Vb?usp=sharing

Cstück Nr. 2

Arsalan Abedian

arsalan.abedian@gmail.com

Abstract. The composition Cstück Nr. 2 (2015) was created using CsoundQt and comprises two principal sound sources. It oscillates between sound and noise, thereby creating a morphing between the timbres and characters of voices and brass instruments, which rise and fall in new sound fields.

Keywords: Granular Synthesis, Spectral Processing, Time Stretching, Voice, Grenze

1 Program note

The text material for the recorded voice in this piece is derived from the German Wikipedia entry on the definition of border. Here an example: “Ein Beispiel für Grenzen von eindimensionalen Räumen ist die „obere“ und „untere Grenze“ in der Mathematik [...]. Umgangssprachlich wird dafür auch Grenzwert, Schwellwert oder Schranke gebraucht.”[1]
The dreamlike (or nightmarish) sound spaces of the spoken word "Grenze", which are created with the help of granular synthesis and time stretching, are presented as sound fields. In these sound spaces, forms emerge and recede, only to reappear in a different form and gestalt. This is similar to the boundaries between countries. In this context, the concept of identity is rendered meaningless. The character of the two border areas is subsumed within a spherical grey zone that simultaneously represents both the borderlands and an independent entity.
The composition Cstück Nr. 2 was created using CsoundQt and features two principal sound sources: brass and voice (recorded voice: Kara Leva). It oscillates between sound and noise, creating a morphing between the sound colours and characters of voices and brass instruments. In this process, the "between", the foreign, can be seen not only as a transition, but also as a new field.
This composition was published on the DEGEM CD 13 (2015), entitled "Grenzen" (Borders) by the German Society for Electroacoustic Music.

2 Biography/CV of Composer, Creator and Performers involved

Arsalan Abedian was born in Tehran, Iran. He commenced his musical studies by learning to play the santur, a Persian traditional instrument, with Omid Sayareh. In 2007, he graduated from Azad University with a Bachelor’s degree in composition, and in 2011, from Tehran University of Art with a Master’s degree in the same field. He proceeded to pursue further studies at the Hanover University of Music, Drama and Media, where he obtained a Master’s degree in Electronic Music in 2014 and a Soloklasse Konzertexamen degree in composition in 2016.
As a composer, organiser and member of the Yarava Music Group, he has participated in numerous concerts, seminars and other events in Iran since 2006. He established the record label Contemporary

[1] Wikipedia contributors. „Grenze“. Wikipedia, The Free Encyclopedia. Available at: https://de.wikipedia.org/w/index.php?title=Grenze&oldid=243614248 (Accessed May 26, 2024).

Music Records (2009) in Tehran. He was a co-initiator of the first competition for electroacoustic music composition in Iran (Reza Korourian Awards) and acted as a jury member and publisher (2016–2018) in this event. As a commissioned composer of several festivals and ensembles, his works have been performed in different countries.

3 Technical notes

The composition of Cstück Nr. 2 employs granular synthesis (*partikkel*) and spectral processing (*pvsmorph* et al.) in CsoundQt. The composition was originally written in stereo.

3.1 Duration

5 minutes and 3 seconds

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

2

Three words by Alejandra

Oscar Pablo Di Liscia

STSEAS Research Program, Escuela Universitaria de Artes, UNQ, Argentina
odiliscia@unq.edu.ar

Abstract. This work is based on three words selected from poems by the Argentine poet Alejandra Pizarnik. The three words are: Errancia, Resolar and Grismente. All these words are *portmanteaux*, that is to say, words that are composed out of several other words using processes of imbrication and combination. There are other sonic materials that are also used in the work, as the poems make explicit reference to them, and their sounds are familiar and very ostensible: water, birds and wind. Sometime, these three elements are combined to produce some kind of abstract sonic landscape while several vocal sounds are presented either in the form of unintelligible sequences or in sequences that expand parts of the three selected words. Was composed entirely using the Csound program, plus several other general tools for mixing and mastering. The author has made available the technical details as well as the *Csound* code of the spatial-spectral granulation resources that he had used in several publications.

Keywords: Computer Music, Granulation, Sound Spatialisation, Sonic Landscape, Sonic Poetry.

1 Program notes

This work is a sort of *electroacoustic poetry-landscape* based on ideas taken from three very similar poems by the Argentine poet Alejandra Pizarnik. In first place, there are three *portmanteaux* words (i.e., words blending the sounds and combining the meanings of others): *Errancia*, *Resolar* and *Grismente*. These three words constitute the basis of the three sections of the work, and are decomposed, time-warped and processed in several ways. In second place, the words are also combined with the sounds of three elements that were also found in the poems: wind, water and birds. The three sections become longer as the work develops, and present the material aforementioned combined in sequences more or less similar, as in a series of variations.

2 Biography/CV of Composer, Creator and Performers involved

Composer and Academic born in Sta. Rosa (La Pampa, Argentina), lives at present in Buenos Aires City. He attained both a Doctoral degree in Humanities and Arts, and a Bachelor Degree in Classical Guitar Performance at Universidad Nacional de Rosario (Argentina) and he also studied composition with the maestros Dante Grela and Francisco Kröpfl. Was Director of the Electroacoustic Composition Program of the Universidad Nacional de Quilmes (UNQ, Argentina), where he is at present Professor of Composition and Computer Music. He was Secretary of Research and Post Graduate Studies of the Universidad Nacional de las Artes (UNA, Buenos Aires, Argentina). He was Director of the "Science and Music" Editorial Collection and Director of the Research Program "Temporal Systems and Spatial Synthesis in the Sonic Art" of the UNQ. At present he is Co-Director of the aforementioned Research Program at UNQ, and Director of a Graduate Program in Digital Arts Sound Applications at UNA. He has published papers and books on aesthetics and techniques

of new music and technologies, as well as developed software for Digital Signal Processing, Musical Analysis, and Composition. His compositions, both electronic and instrumental, have been recognized by awards both nationally and internationally, and have been recorded, edited, and performed in several countries (Argentina, Chile, Uruguay, Cuba, USA, France, Spain, México and Holland).

3 Technical notes

The work was entirely generated using the *Csound* program, by processing the source materials with a spectral-spatial granulation environment programmed by the composer [1], [2]. As in other works by the author, several interacting strategies between the synthesis/transformation processes and the spatialisation are explored to produce paradoxical results both in the perceptual grouping of sonic events and in the rooms acoustics. The spatialisation was accomplished applying the *Ambisonics* technique by means of *Csound* instruments and *UDOs* (*user-defined opcodes*) developed by the author using several *Csound* resources and Opcodes. Both the direct sound sources and the effect of room acoustics (early and late reverberation) were synthesized and the author has made available the detailed procedures as well as the corresponding *Csound* code in several publications.

3.1 Duration

C.A. 6 mins., 44 secs.

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

The optimal way of reproducing this work is in a 3D loudspeaker rig. The original encoded file of the work, in 3th Order Ambisonics (16 encoded audio channels) B-Format ACN channel Order, SN3D Normalization may be provided by the author to be properly decoded by the technical staff so as to use the 3D reproduction system available in the optimal way. Otherwise, if requested by the technical staff, the author may provide the needed channels already decoded with their placement in order to use the full system specified in the technical guidelines.

3.4 Link

https://drive.google.com/file/d/1oO3MgJOZuEh7GCpxx3pWC0DSrnrnxJqv/view?usp=sharing

References

1. https://github.com/ odiliscia/the_grainer_Csound_gh
2. Di Liscia, O. P.: Spectral and 3D spatial Granular synthesis in Csound. In: ICSC 2017 Proceedings, pp-47-53. Montevideo (2017).

Oscillation Of Life

Jan Jacob Hofmann

jjh@sonicarchitecture.de

Abstract. The piece is spatially encoded in 7th order Ambisonic. The sounds were generated exclusively with the sound synthesis program "Csound" and the editor for composition "blue" to create the piece. Also, the spatialisation in 7thth Order Ambisonic was done via my own code using Csound within my blue-environment for spatialisation. The piece is about the generating forces of nature.

Keywords: Higher order Ambisonics, Csound, blue, Electroacoustic Music

1 Program notes

This piece is about the generating forces of nature. To be more precise, it is about the idea of an underlying universal power that gives shape and energy to all living beings. What if there was a yet undiscovered oscillating energy beyond acoustic and electromagnetic oscillation, that gave shape, energy and interconnection to all living beings? That enabled/guided/facilitated the organisation of molecules and cells to higher organisms, beyond genetic chemical reactions and metabolism, opposed to the common increase of entropy? That creates shape like symmetry up to far more complex mathematical order, beauty out of chaos by transmitting harmonic information? What would that oscillation sound like, if we could perceive it? Would we listen? Would we be able to tune in?

2 Biography/CV of Composer, Creator and Performers involved

Jan Jacob Hofmann was born 1966 in Germany. Diploma, branch of architecture at the Fachhochschule Frankfurt am Main, University of Applied Sciences in 1995. Entered the class of Peter Cook and Enric Miralles at the Staedelschule Art School Frankfurt am Main in 1995, a postgraduate class of conceptual design and architecture. Diploma at the Staedelschule in 1997. Works as a composer, photographer and architect since.
Since 1986 dealing with composition and electronic music. Music for performances. Since April 2000: Work on spatialisation of sound. Several international performances in America, Europe and Asia since.
Research on Ambisonic and other spatialisation techniques. Development and publication of Csound based tools for spatialisation via higher order Ambisonic.
Became associate researcher in summer 2005 at the “Signal Processing Applications Research Group”, University of Derby, England. Member of the board of the German electroacoustic music society (DEGEM) from 2006 to 2022.

3 Technical notes

The piece is spatially encoded in 7th order Ambisonic. The sounds were generated exclusively with the sound synthesis program "Csound" and the editor for composition "blue" to create the piece. Also, the spatialisation in 7thth Order Ambisonic was done via my own code using Csound within my blue-environment for spatialisation. An additional program used was "Cmask" for the generation of stochastic events and patterns within blue.
The sound sources are purely synthetic, mostly simple sinewaves altered by Chebyshev polynomial distortion and modulation.
The extension of my blue-environment for spatialisation from 3rd to 7th order Ambisonic has been kindly supported by the Musikfonds Deutschland.

3.1 Duration

10:44 min.

3.2 Category

Electroacoustic music on fxed medium

3.3 Channels

The number of playback channels used will be 21.2. A decode in 3rd order Ambisionic tailored to the present layout and number of speakers will be provided by the author. Alternately the 7th order encoded file can be transmitted for a decode on site. The sampling-rate is 48 khz/24bit.

3.4 Link

Link to a lossless compressed recording of the piece in FLAC format:
http://www.sonicarchitecture.de/downloads/OscillationOfLife/OscillationOfLife3rd-binauralstereo.fac
The multichannel piece has been decoded to a 3rd-order-binaural-stereo downmix using 22 virtual speakers.

References

1. SonicArchitecture-site: http://www.sonicarchitecture.de

Gendy Cloud

Serkan Sevilgen

Istanbul Technical University (MIAM)
ssevilgen@gmail.com

Abstract. The "Gendy Cloud" (2022) is a real-time, networked, multichannel music piece performed by WORC, a telematic ensemble. Performers control their instruments remotely via a web interface, manipulating one or more instances of a Csound-based software instrument based on Xenakis's GENDYN algorithm. Inspired by "Xenakis22: The Centenary Symposium," this project commemorates Xenakis, enabling collaborative music-making across distances. The piece was performed at Xenakis Networked Music Marathon (Athens, 2022), Sonified Symposium (Istanbul, 2022) and presented as a demo at the Internet of Sounds Symposium (Pisa, 2023), also showcased on the Csound website.

Keywords: networked music, telematic performance, multichannel audio, web interface

1 Program notes

The "Gendy Cloud" (2022)[1] is a networked, multichannel music piece that will be realized in real time by *WORC*, a telematic ensemble. The ensemble members could control their instruments remotely via a web interface. Any performer can control one or more instances of the software instrument based on Csound implementation of Xenakis's GENDYN algorithm. The control parameters are limited to reduce the learning curve and increase the adaptability to the existing interfaces. However, use of stochastic processes in the instrument allows performers to create varied timbre, patterns, and textures in a multichannel diffusion system.

The project was inspired by an event during "Xenakis22: The Centenary Symposium"[2]. Orestis Karamanlis[3] utilized GENDYN (a dynamic stochastic sound synthesis algorithm conceived by Iannis Xenakis) and prepared an audio stream that conference participants can use on their mobile phones to hear in the front of the building where Iannis Xenakis was wounded. It was a touching moment that we could be able to commemorate a great composer through his work. The idea arose from the event that if it is possible to build a software instrument based on the GENDYN algorithm that leads to collaborative music-making regardless of the physical locations.

2 Biography/CV of Composer, Creator and Performers involved

Serkan Sevilgen is an electroacoustic music composer who employs his professional programming skills to create computer music. Sevilgen has an MA in Sonic Arts from the Center for Advanced Studies in Music (MIAM) at the Istanbul Technical University. He is a co-founder of Soundinit, an

[1]https://csound.com/showcase/2023/03/18/Gendy_Cloud
[2]https://xenakis2022.uoa.gr/
[3]https://orestiskaramanlis.net

initiative focusing on sound, and a founder of *WORC*, a networked music ensemble. He is a member of the Istanbul Coding Ensemble. His musical works and research focus on computer music, sonification, networked music systems, web audio, live coding, stochastic procedures, and soundscape. Sevilgen has presented his works at various international events such as ICMC, NIME, SMC, ISMIR, Xenakis Networked Performance Marathon, and the New York City Electroacoustic Music Festival.
https://serkansevilgen.com/docs/Serkan-Sevilgen-CV.pdf

WORC ensemble invites computer music practitioners to their events to perform.

3 Technical notes

In The Gendy Cloud has three components:

- A computer at the concert venue hosts a Csound-based software instrument: The instrument comprises Gendy opcode based on Iannis Xenakis's GENDYN algorithm.
- Web Interface as a controller: Performers that can join from any geographical location.
- Remote OSC: A client-server library for delivering control messages over the Internet.

3.1 Duration

8 minutes

3.2 Category

Live-Electronics

3.3 Channels

The piece utilizes higher order ambisonics (up to 7th order) and has capability to conform to any speaker layout that organization offers, e.g 21.2

3.4 Links

Demo of the project
https://www.youtube.com/watch?v=QDBQICae23A

SONIFIED Symposium
Arter Museum (Istanbul, Turkey), 29 December 2022
https://www.youtube.com/watch?v=uBC3avitRGg

WORC Ensemble members:
– Danny Fratina (Boston, USA)
– Umut Eldem (Antwerp, Belgium)
– Dimitri Papageorgiou (Thessaloniki, Greece)

– Vasilis Agiomyrgianakis (Athens, Greece)
– Nihan Tahtaişleyen (Istanbul, Turkey)
– Manolis Ekmektsoglou (Istanbul, Turkey)
– Serkan Sevilgen (Istanbul, Turkey)

XNPM22: Xenakis Networked Performance Marathon
Athens Conservatory (Athens, Greece), 17 December 2022
https://www.youtube.com/watch?v=bKlwMrMdlnI

WORC Ensemble members:
– Iannis Zannos(Japan)
– Vasilis Agiomyrgianakis (Greece)
– Serkan Sevilgen (Greece)
– Manolis Ekmektsoglou (Turkey)
– Nihan Tahtaişleyen (Turkey)

Traverse: for Recorder and Electronics (2024)

Bethanie Liu

Berklee College of Music
bliu3@berklee.edu

Abstract. 'Traverse: for Recorder and Electronics' (2024) is an eight-minute electroacoustic composition for acoustic recorders and electronics. All sounds are created through live improvisational melodies performed by the composer on soprano and alto recorders, then processed using a range of Csound and Cabbage plugins, including custom-made Cabbage plugins by the composer and those from the McCurdy Collection. The composer also performed over the processed tape, exploring the interaction between acoustic and electronic sounds through counterpoint between live recorder and processed sounds. This piece depicts the composer's journey of walking away from the scars of trauma. The electronic sounds, derived from processing the recorder performance, convey memory flashbacks and swirling emotional disorientation. The performance version of this composition will feature live recorder playing and live effects processing.

Keywords: Csound, Cabbage, Recorder, Electronics, Electroacoustic Composition

1 Program notes

'Traverse: for Recorder and Electronics' (2024) is an eight-minute electroacoustic composition for acoustic recorders and electronics. The composer recorded her live improvisations on soprano and alto recorders, then processed the recording with a range of Csound and Cabbage plugins, including her custom-made Cabbage plugins and those from the McCurdy Collection. This piece depicts the composer's journey of walking away from the scars of trauma. In this piece, acoustic recorders and electronics echo and interact with each other to convey intertwining memories of the past. The dark atmospheric drones and brash ring modulation sounds, all created by processing the composer's recorder performance in Csound, depicts a state of lostness and confusion. Contemporary extended techniques for the recorder, such as flutter tongue and sputato are also featured in the improvisational melodies. The piece eventually resolves back to the theme, depicting the composer's return to the same place after years, still agitated, but learning to be at peace with the past.

2 Biography of the Composer/Performer

Bethanie Liu is an electroacoustic composer, electronic performer and researcher. She is also a classically-trained multi-instrumentalist, having received training in piano, flute, organ and recorder from a young age. She has performed worldwide as a featured recorder soloist at esteemed venues like Carnegie Hall(US), Conservatorium van Amsterdam and St John's Smith Square(UK). She now extends her virtuosity into the electronic realm, blending acoustic instruments with contemporary electronic styles.

3 Technical notes

All sounds are created with live improvisational melodies performed by the composer on soprano and alto recorders, featuring contemporary extended techniques for the recorder such as flutter tongue and sputato. The recorder performance is then processed with a range of Csound and Cabbage plugins, including Shredulator, BreakBeatCutter, SpectralDelay, MultitapDelay from the McCurdy Collection. The composer has also created her own reverb, delay, ring modulation effects plugins in Cabbage and had used them to process her recorder performance for this composition.

3.1 Duration

The piece is 8 minutes and 26 seconds long.

3.2 Category

Mixed pieces for instrument(s) and electronics. (fixed medium).
The composition is for Recorder and Electronics.
The composer is the performer of the piece, and will attend the conference to perform it live if accepted.

3.3 Channels

8 channels (circle 8) will be required. The submission is a stereo downmix as requested in the submission guidelines.

3.4 Link

https://drive.google.com/file/d/1Gmaxvmyo_ZVPUJVcTJP80fycWTuclgz-/view?usp=sharing

References

1. Lazzarini, V.: Spectral Music Design. Oxford University Press (2021)
2. Boulanger, R.: The Csound Book. MIT Press, Cambridge (2000)
3. Csound site, http://csound.com
4. Ian McCurdy site, Catalogue of Example Csound Files, iainmccurdy.org/csound.html
5. Csound Manual site, https://csound.com/manual.html
6. Cabbage Audio site, https://cabbageaudio.com/

Caibleadh

Shane Byrne

Technological University of the Shannon
shane.byrne@tus.ie

Abstract. "Caibleadh, voices you hear in the distance at sea..especially on a calm night in the mouth of the bay...They say not everyone can hear these things, but you have to be there at the right time".

Keywords: Electroacoustic, Fixed Media, Multichannel

1 Program notes

The Last Battle of Mag Tuired was fought between the Tuatha De Dannan, an ancient race of ancient Irish dieties, and their enemies, a supernatural people known as the Fomori. The leader of the Fomori, Balór na Súile Nimhe, was defeated in battle by the hero Lugh Lámhfada, resulting in the Fomorian army being cast into the depths of the sea off the coast of Ireland.
Haunting voice-like calls heard in the distance across the water on still nights, known as cailbleadh, are said to be the songs of the lost Formorian spirits, exiled to beneath the waves.
The idea of cailbleadh came to mind when listening to seals along the coast, their calls echoing across the cliffs and resonating in the caves, creating an almost preternatural soundscape.

2 Biography of the Composer, Creator and Performers involved

Shane Byrne is a composer and educator working in the field of creative media. His main area of interest is electronic music composition and in particular, the ways in which embodied music cognition and generative systems can influence the creative process.
His works have been performed internationally at festivals and conferences including ICMC, TIES, ISSTC, MUSA, SMC, iFIMPac, Sonorities, and KLG.
He is currently the program lead on the BSc (Hons) in Music and Sound Engineering at TUS Midlands where he lectures in Electroacoustic Composition, Interactive Audio, Audiovisual Composition, Visual Creation, and Audio Electronics.

3 Technical notes

The source material used in this piece is comprised of field recordings all gathered along the Irish coast. A large body of the material used in the piece was collected using a collection of homemade hydrophones.
Variations of the material were created using a generative playback system built using Csound. Subsequent transformations were created using a variety of granular processes and then finally arranged in Abelton Live.

Shane Byrne

3.1 Duration

7 minutes 54 seconds

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

Octophonic 8.1.

3.4 Link

https://drive.google.com/file/d/1crAXlP85ikGmBqaxp7OQKy7XCOKz4dxO/view

REEHD

Clemens von Reusner

Independent Composer
info@cvr-net.de

Abstract. REEHD is not based on sounds of real instruments, but on sounds generated by physical modeling. Physical modeling allows to go beyond the limits imposed by real instruments as well as the limits imposed by human players. This can result in certain sounds no longer having any relation to known instrumental sounds. In REEHD sound objects interact as sound gestures as well as textures in a concept of composed spatial counterpoints in virtual spaces.

"But no one should be afraid that looking at signs leads us away from things; on the contrary, it leads us into the innermost of things." (Gottfried Wilhelm Leibniz, 1646-1716)

Keywords: physical modeling, ambisonic, immersive

1 Program notes

REEHD is not based on sounds of real instruments, but on sounds generated by physical modeling. Physical modeling allows to go beyond the limits imposed by real instruments as well as the limits imposed by human players. This can result in certain sounds no longer having any relation to known instrumental sounds. In REEHD sound objects interact as sound gestures as well as textures in a concept of composed spatial counterpoints in virtual spaces.

"But no one should be afraid that looking at signs leads us away from things; on the contrary, it leads us into the innermost of things."

(Gottfried Wilhelm Leibniz, 1646-1716)

2 Biography/CV of Composer, Creator and Performers involved

The composition of the sound itself and its arrangement and movement on individual paths in the virtual acoustic spaces of multi-channel loudspeaker configurations are the center of the electroacoustic works and radiophonic audio pieces by german composer Clemens von Reusner (born 1957).

In his sound language, he sometimes also refers to contemporary and historical works from music, literature and the visual arts. At the end of the 1980s, he developed the music software KANDINSKY MUSIC PAINTER, which uses graphic tools to create musical structures via MIDI.

Clemens von Reusner is a member of the Academy of German Music Authors and was nominated for the GEMA German Music Author's Award in 2023. His works have been awarded national and international prizes, including the 2024 Thomas Seelig Fixed Media Prize from the German Society

for Electroacoustic Music (DEGEM) for his complete works. They are performed at renowned international festivals for contemporary music in Asia, Europe, North and South America. Clemens von Reusner received invitations to the World Music Days for New Music 2011 in Zagreb, 2017 in Vancouver and 2019 in Tallinn.
www.cvr-net.de

3 Technical notes

Csound plays an important role in my workflow as a composer of electroacoustic music. Csound is – amongst other software like SOX, CDP, SuperCollider and REAPER (DAW) – an indispensable tool for the manifold tasks of soundprocessing of the sounds in REEHD.

3.1 Duration

07:11

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

8

3.4 Link

https://we.tl/t-nphKxmbeiy

Link to 8-channel version
https://we.tl/t-nzsVr5rHvm

Eleven Questions (2024)

John ffitch

Alta Sounds
jpff@codemist.co.uk

Abstract. An 8-channel real-time internet duet in which one performer is playing on stage in the concert hall and the other is performing from his home studio and appears to the audience over Zoom. All audio is synthesized and no samples are used. In the 8-minute piece, the computer plays an active role in synthesizing and spatializing chord progressions that move from 59-tone per octave at the beginning to 12-tone per octave by the end. ASCII keyboards serve as the controllers and triggers and text messages appear on the screen of each performer to indicate how one or the other might be changing the register, speed, chord progressions, transpositions, and timbres. Each player assumes differnt roles during the course of the performance – sometimes soloist, other times accompanist, sometimes conductor, sometimes house mix engineer. Both are hearing what each other is doing in each location and moderating thier roles in response to the evolution of the piece and the chords generated by the computer.

Keywords: Real-Time, Duet, Synthetic, Algorithmic, Generative, Microtonal, Web-based, ASCII.

1 Program notes

'Eleven Questions' (2024) is an 8-channel internet-duet with an 'ensemble' consisting of 4 'generative' computer players (the 'choir"), and 2 live ASCII players – one playing on stage in the concert hall and the other playing remotely over the WEB via OSC and ZeroTier. The remote player is projected into the concert hall via ZOOM. The live coding of the on-stage performer is projected onto another screen. Both are hearing the entire work as it is all being realized in real-time; both are sending and receiving 'text-print' messages as feedback informing each other about what motives (questions) they are selecting, what transpositions and tempi they are setting, what chords and timbres they are playing, and how they might be affecting the sounds of the computer players and each other. Over the course of the 7 minute piece, the 'tunings' of the computer harmonies and the melodic motives move from 59-tone to 12-tone. Each motivic 'question' and every note from the 'choir' comes from a discrete location and the live performers have complete control over the timing, the tempo, the register, the dynamics, and the overall mix of all the elements in the piece. As they listen to each other, and to the computer, they question and answer, accompany and lead, compliment and contradict; in some ways, "Eleven Questions" could be considered a structured internet Csound jam as it is never exactly the same, but the players are all always 'reading' from the same algorithmic 'lead-sheet'.

2 Biography/CV of Composer, Creator and Performers involved

Composer and Web Performer - John ffitch was definitely born after WWII, in that part of the United Kingdom which is God's own county, certainly educated at an East Anglian university in

the sixties, and despite his long hair and lengthening beard, and the uncertain spelling of his name, was never a hippie. His entire professional career has been as an academic mathematician/computer scientist, and for most of that time he has held the Chair of Software Engineering at Bath, a subject about which he knows little. His main interests have been in Relativity, Planetary Astronomy, Computer Algebra and LISP, but he has been known to dabble widely, for example in tank warfare, Latin poetry, Arabic linguistics, compilers, and company management, all with some lack of success. Strangely enough, he won the Adams Prize for Mathematics more than 25 years ago. Hobbies include maintaining Csound, receiving e-mail, and complaining about the Web.

Live Concert Performer – Richard Boulanger is an Adjunct Professor of Electronic Production and Design at the Berklee College of Music in Boston and has been collaborating and performing with John ffitch since 1987. This work is the product of their weekly jam sessions, over Zoom, from their home studios in Bath, England and Dighton, Massachusetts (New England).

3 Technical notes

The real-time interactive-generative-algorithmic piece features FM, Waveshaping, Scanned, Subtractive, and Additive Synthesis with some Csound effects all rendered in the concert hall from the 'concert' player's laptop output from his 8-channel MOTU (or Behringer) audio interface. (If needed, eight short TRS to TRS cables (or TRS to XLR cables) – could be provided by the concert performer to patch into a house snake.) For the performance, there is a need to project onto a large screen from the live player's laptop – HDMI out (which will show the text that appears on his screen as he performs, and will also show the other player Zooming in on the Web and see him performing as well.)

3.1 Duration

The duration of the work is EIGHT minutes.

3.2 Category

Live-Performance (Internet-based)

3.3 Channels

Best when performed on 8-channels with subWoffers, but it could also be performed on 4 or 2-channels with SubWoffers if that is all that is available in the specific hall-venue.

3.4 Link

https://www.dropbox.com/s/yu4erntuwfjj09c/ICSC2024-Submission-11questions-2takes.zip?dl=0

Decay

Patrick Dunne

Technological University of the Shannon
Padd.dunne@gmail.com

Abstract. Decay is a short electro-acoustic audio/visual piece inspired by the work of Jonty Harrison. The composition makes use of AI-generated visuals, serving as an interperative visual score for the audio. The audio content is comprised of recordings of matches being lit, extinguished and broken, with some additional nature sounds. The piece makes extensive use of the Grain3FilePlayer granular synth written by Iain McCurdy in Csound.

Keywords: AI, Electro-acoustic, composition

1 Program notes

Decay is inspred by the Works of Jonty Harrison, particularly Surface Tension and EQ. The goal of the piece was to create an entire composition using a single sound source – in this case, a box of matches. The visuals were generated using Runway's text-to-video and video-to-video AI tools. The visuals were further manipulated using the Runway motion brush to distort the generated images creating abstract shapes in the process.

2 Biography/CV of Composer, Creator and Performers involved

Masters by Research student at Technological University of the Shannon, Ireland.

3 Technical notes

The audio content of Decay is comprised of recordings of matches being lit, extinguished and broken. The rercordings were manipulated using two granular synths – Portal by Out- put and Grain3FilePlayer, written by Iain McCurdy in Csound. The results were edited and mixed in ProTools using the AI generated visuals as an interperative score to dictate the structure, tone and movement of the piece.

3.1 Duration

1 minute 34 seconds

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

2.

3.4 Link

Decay.mp4

Studio VII

Roberto Doati

info@robertodoati.com

Abstract. My *Studi I-VIII* are inspired by Karlheinz Stockhausen's *Klavierstücke I-VIII*. If *Klavierstücke I-IV* (1952-53) represent a sort of sketches of the electronic pieces to come, *Klavierstücke V-VIII* (1954-55) reveal a new attention to time which at the same time 'stretch' the form according to "statistical form criteria" and allows the author to build different timbres. In my studies I wanted to recreate the colour of those years' electronic sounds. The many different timbres are obtained with modal synthesis applied to audio signals produced by a set of Julia (implemented in CSound by Hans Mikelson, 1999). Each sound is conceived as a *momentform*.

Keywords: synthesis with fractals, modal synthesis, Elektronische Musik, Stockhausen.

1 Program notes

My *Studi I-VIII* are inspired by Karlheinz Stockhausen's *Klavierstücke I-VIII*. These piano works revolve around the electronic experience of *Elektronische Studie I* and *II*. If *Klavierstücke I-IV* (1952-53) represent a sort of sketches of the electronic pieces to come, *Klavierstücke V-VIII* (1954-55) reveal a new attention to time which at the same time 'stretch' the form according to "statistical form criteria" and allows the author to build different timbres that emerge from the constant use of resonances produced by the silent pressure of the keys. In my studies I wanted to recreate the colour of those years' electronic sounds, especially in its main morphology, very similar to that of piano sounds, and strongly correlated to the spectrum obtained with physical models applied to audio signals produced by a set of Julia [1]. *Studio VII* is structured as if it were a sketch of *Klavierstücke VII*. It follows its dynamics and density using three morphological typologies: fast arpeggios, long single sounds, slow arpeggios. Each sound is conceived as a *momentform*.

2 Biography

Studied Electronic music with Albert Mayr and Pietro Grossi at the Firenze Music Conservatory, then in Venezia with Alvise Vidolin. From 1979 until 1989 he has been working as a composer and researcher at the Centro di Sonologia Computazionale, University of Padova. From 1983 to 1993 he was a staff member of L.I.M.B. (La Biennale di Venezia). His teaching career as Professor of Electronic Music ended up in 2020 at the Music Conservatory "Giuseppe Nicolini" in Piacenza. Fellow and composer in residence in several places such as Centre de Recherches et de Formation Musicales de Wallonie in Liège, Fondazione Bogliasco, Rockefeller Foundation, MacDowell Colony. Since 2013 he produces audiovisual works on the aesthetics of food such as *Seppie senz'osso*, *Il suono bianco*, *Il suono rosso*. His last compositons are included in *The Spirit of Risk*, a concert-hommage to Anthony Braxton for alto saxophone and live electronics.

3 Technical notes

All the sounds are synthesized from multiple modal filtering (*mode*) of Julia Set signals [1]. Some samples from Stockhausen's instrumental music are filtered using *streson* and *butterbp* and added to the decay of some sounds. The arpeggios are obtained with random distribution applied to *event_i* entry delay and duration. The global reverberation is a convolution (*pconvolve*) with IR I recorded from an EMT 140 Plate Reverberator. The composition is built with two layers of the same .orc and .sco: one at 60 mm, one at different mm (50/71/57/63.5/40).

3.1 Duration

7 minutes

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

Octophonic 8.1 system

3.4 Link

https://drive.google.com/drive/folders/1H0STciIX42zXbjcB8IyruSnk-jjZit8X?usp=sharing

References

1. Mikelson, H.: Sound Generation with the Julia Set. Csound Magazine Summer 1999

Woodland Understorey

Mark Ferguson

Independent (No Institution)
markfergusonaudio@gmail.com

Abstract. A scene from the Cotswolds (UK), composed as a sonic cross-section of a broadleaf woodland habitat. Processing for the piece relied heavily on random modulation of *grain* opcode parameters, with Csound used as a kind of granular 'rain generator' to augment and build upon existing field recordings of precipitation. Rain-shower width was adjusted via random modulation of the *pan2* opcode. Towards the end of the piece, *hrtfstat* was used to position a small stream binaurally, along the composed forest floor (audible from 04:10 onwards).

Keywords: wildlife; habitats; woodland; rain; Csound; granular synthesis; nature; soundscape

1 Program notes

Recollections from a Cotswold woodland. Tall ash and sycamore trees in fog, heavy with condensation; leaves bending, thick drops rolling off them as a kind of half-rain. Tawny owls and pheasants, louder than expected. An evening shower moves through. It is a scene of shelter and delicate interplay, infused with the smells of damp earth.

2 Biography/CV of Composer, Creator and Performers involved

Mark Ferguson is a UK-based wildlife sound recordist and sound artist.
Noted for his introspective documentary style, his work is influenced by a diversity of subjects such as nature conservation, sonic archiving, electroacoustic composition and video gaming. Much of his current practice is inspired by culturally misunderstood species, and the unique power of audio technologies to draw attention to their stories.
Mark's award-winning recordings and projects have been broadcast by the BBC, mentioned by the Guardian, and featured in leading arts and cultural venues worldwide. He is a member of the Wildlife Sound Recording Society, Bat Conservation Trust and Wildfowl & Wetlands Trust, and is an established contributor to the Wildlife and Environmental Sound Archives at the British Library.
linktr.ee/fergusonic

3 Technical notes

In this this piece (a re-composed memory of a recording experience in a Cotswold woodland), Csound was used primarily as a 'rain generator', to add layers of precipitation over original field recordings. Samples of individual raindrops, as well as sequences of rainfall, were used as source materials for the *grain* opcode. This opcode was modulated by a complex series of *randomi* opcodes, providing variations to grain density, pitch, etc. The *pan2* opcode was also modulated via *randomi*, to scatter rainfall across the stereo field. Gaussian and hamming windows were specified using *ftgen* and applied

to *grain* as needed, with rendered results selected for further DAW manipulation. The small stream emerging at the end of the composition was positioned binaurally, using the *hrtfstat* opcode (note that this binaural placement still functions well over 2.1 loudspeaker systems). Field recordings used in the piece were gathered in Bisley, in the heart of the English Cotswolds. The recording of the small stream at the end of the piece was made in the Sperrin Mountains, in Northern Ireland.

3.1 Duration

04:38

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

Stereo 2.1

3.4 Link

https://drive.google.com/file/d/1bwlrcmR3akAL-Ao9NH3Q9V31ZqgLFkY6/view?usp=drive_link

“Franz Strauss – Five Etudes” (2021) for natural horn and electronics

Tarmo Johannes

trmjhnns@gmail.com

Abstract. “Franz Strauss – Five Etudes” is a composition for natural horn and live electronics using Csound. The piece places in contrast very vulnerable pracitcing situation of a beginner horn player and noisy, merciless intervention of the electronics, the uttermost predictability of the etudes and the unexpectedness created by computer algorithm.

Keywords: Csound, natural horn, live electronics, algorithm

1 Program notes

Tarmo Johannes on "Franz Strauss – Five Etudes" (2021): "I created this piece in the summer of 2021 when Erik Alalooga, an Estonian noise artist invited me to play at a open air summer experimental music event in Tallinn. At that time, I had relatively recently started learning the natural horn as a new hobby. Considering Erik Alalooga's preference for rather harsh sounds, I wanted to combine my especially novice attempts at playing Franz Strauss's horn etudes with electronic processing, which, according to a certain algorithm, mercilessly overrides the horn's triadic passages from time to time. Additionally, there is a contrast here between the perhaps somewhat tedious regularity typical of etudes and the unpredictability of the processing."

2 Biography/CV of Composer, Creator and Performers involved

Tarmo Johannes (1976, Tallinn) is an Estonian flutist dedicated primarily to performing contemporary music. He is founder and leader of ensembles U: (flute-clarinet-violin-cello-piano www.uuu.ee) and Resonabilis (voice-flute-cello-kannel www.resonabilis.com), member and has played a number of concerts of solo flute repertoire with great success. Since 2019 he is member of the Estonian Electronic Music Ensemble.
In recent years Tarmo Johannes has been actively engaged also with sound synthesis and programming. His main interest is creating algorithmically controlled pieces for live performance, often with audience’s participation. Since 2015 Tarmo Johannes is the main devloper of CsoundQt. Tarmo Johannes teaches flute at Tallinn Music and Ballet School (Estonia) and different subjects related to new music at the Esonian Academy of Music and Theatre.
http://tarmo.uuu.ee/

3 Technical notes

The composition uses live natural horn playing as source, Csound uses various distortion opcodes to treat the recorded sound. The piece consist of purely acoustic sections and “noise windows”. How long, at what time and of which content are the noise windows is up to Csound to decide and can be a surprise also for the player.

Tarmo Johannes

Output: stereo, from audio interface (TRS/XLR)
The horn is playing unamplified, the output from Csound should be fairly loud, to cover the acoustic horn almost completely.
I will bring: computer, audio interface, microphone + cable
I need: stereo amplification, cables from sound interface to the main desk.

3.1 Duration

7:30

3.2 Category

Mixed pieces for instruments(s) and electronics (fixed medium and/or interactive)

3.3 Channels

2

3.4 Link

https://drive.google.com/file/d/1MJODQGN2KZXGMCIcdRnjWZF6tAWj7Ctg/view?usp=sharing

Appendix

Included in zip file:
- Csound program horn-noise.csd included. To be run in CsoundQt.
- Photo of Tarmo Johannes playing natural horn. Photo: Rene Jakobson

A fashionable nightclub

Jean-Basile Sosa

Independent
sosa.jeanbasile@gmail.com

Abstract. A fashionable nightclub is a live electronic music performance spatialized on variable loudspeaker arrays. With this immersive creation, Jean-Basile Sosa delivers an ethereal, phantasmatic version of some of electronic musics played in American nightclubs in the 80s and 90s...

Keywords: sound, electronic, idm

1 Program notes

A fashionable nightclub is a live electronic music performance spatialized on variable loudspeaker arrays. With this immersive creation, Jean-Basile Sosa delivers an ethereal, phantasmatic version of some of electronic musics played in American nightclubs in the 80s and 90s...
It's also a reminiscence of certain spaces of social, collective and individual freedom, where marginal cultures unfold, often foreshadowing the mores and habits of tomorrow...
Without ever falling into a parody of house music or techno, the project nevertheless assimilates some of the most significant characteristics of these popular musical currents: the repetition and obstinacy of the pulse, the complete abstraction of the electronic sonorities used, the regular periodicity of squares, phrases and durations, the intuitive memorization of harmonic and rhythmic cycles and loops...
The project is also motivated by the ongoing development of a digital environment dedicated to musical performance and sound spatialization. Transmissible and perennial, this environment should ideally adapt to all types of audio broadcast configuration, from projected stereo to the most modern three-dimensional sound spatialization techniques such as ambisonics.

2 Biography/CV of Composer, Creator and Performers involved

Jean-Basile Sosa's work explores the diversity of electroacoustic creation, ranging from acousmatic art to mixed and electronic music, from music for film and performing arts to sound installations and audio-visual art objects.
His music has been programmed in France and abroad (festivals Mixtur, Exhibitronic, Détours de Babel, ICSC Csound, SMC, États généraux du film documentaire, NYCEMF, Musiques En Scène, CIMXX, Futura, Le Mans Sonore).
With a degree in Musicology, Jean-Basile Sosa also holds a Master's degree in Electroacoustic Composition from the CNSMD in Lyon, where for five years he received a singular vision of the discipline from the angle of aesthetics, art history and computer music.

3 Technical notes

A fashionable nightclub is a pure elecrtonic piece, without the manipulation of sound recording. The "step-by-step" sequencing that characterizes much of popular electronic musics, such as house and techno, is revisited in this project thanks to the use of two programming languages in particular: firstly, for the production of all of original electronic sounds and all the sound synthesis engines, Csound. Secondly, Antescofo, a high-level scripting language for organizing, managing and writing data in real time.
Finally, as a graphical interface and midi communication medium, MaxMsp lets you use physical controllers to play the full range of instruments and tools developed specifically for this purpose.

3.1 Duration

As it is a live performance, the duration can be variable but ranged to 10 to 13 minutes max. For ICSC, I will try to not exceed 11 minutes.

3.2 Category

Live-Electronics

3.3 Channels

In absolute terms, the performance spatialization device should ideally adapt to all types of audio broadcast configuration, from projected stereo to the most modern three-dimensional sound spatialization techniques such as ambisonics.
Depending on the rehearsal and preparation time in situ, the peerformance spatialization will be adapted to the larger available configuration (21.2).
We should say at this time that an octophonic network of loudspeakers will be used as a minimum. However, we do not rule out using the 21.2 main device as an acousmonium with projected stereo sound technique...

3.4 Link

https://drive.google.com/drive/folders/11_wAbqBqS8tcqg5ghjAN9nTGiFxGG4H9?usp=sharing

Sievert

Jinhao Han

Sichuan Conservatory of Music
Viktor_H@qq.com/hanjinhaoworks@gmail.com

Abstract. Sievert is an audio-visual music for Csound and GLSL Shader, inspired by the decay in nuclear physics, Sievert is the international unit of absorbed dose of nuclear radiation, meaning that this work was created using the radiation values of uranium ore picked up by GM counters. The work was created using Csound, ATS, Magic--the front-end graphical interface of the GLSL Shader, an audio-visual music created through the synthesis and decomposition of sound.

Keywords: ATS, GLSL Shader, Audio-Visual

1 Program notes

This work is inspired by nuclear decay, using Csound and sound analysis-resynthesis measures to construct an audio-visual electronic music that displays the element decay within the theoretical framework of nuclear physics. In this piece, taking the decay process of Uranium-235 to Lead-207 as an example, radioactive nuclides release a large amount of energy through a series of α and β decays, reducing their own entropy to reach a relatively stable state. In this decay process, unstable elements lower their energy by emitting high-energy particles, gradually leaving the excited state to become a stable element. This reflects my perspective of viewing the development and change of the world from a microscopic viewpoint, extending the idea that "decay" is a process in which high-energy matter gradually stabilizes through a series of destructive changes to stabilize itself, shedding uncontrollable parts, and ultimately forming a new individual with a tight and regularly stable structure.

2 Biography/CV of Composer, Creator and Performers involved

Jinhao Han (1998), postgraduate student of New Media Music, Department of Electronic Music, Sichuan Conservatory of Music, under the tutor Wanjun Yang, and will be graduating in 2024 and currently looking for a PhD institution. Majoring in new media music and coded music composition. He has performed his work at ICMC, IEMC, ICSC, and NY-CEMF several times from 2018-2024, and has published several papers in Music Programming Languages and Electronic Music Education in 2019-2024.

3 Technical notes

This work is an audio-visual electronic music based on coding synthesis and GLSL graphic generator, utilizing sound processing programs written in Csound to interact with Geiger counter data, transforming the counter data into control data for sound synthesis and processing. In this work, the

Jinhao Han

Geiger counter's data is recorded in an audio file, with each count corresponding to a pulse signal. Peak detection through the real-time synthesis control language allows for the extraction of pulses from the audio file, which are then converted into Sievert values through a timer and Geiger algorithm. Based on the number of pulses and the value of the conversion., Sievert values are mapped to the intensity of sound synthesis, the order of overtones, and the starting order of overtones; in Csound, Sievert values are mapped to the sequence number of DSP function groups and output volume. By means of a predefined set of event_i, the method of sound synthesis and waveform envelope are controlled, and Sievert values and audio data are packaged into image data that can be read by GLSL, using the output Sievert values and audio data for the generation and change of graphics, constituting a real-time audio-visual electronic music piece.

3.1 Duration

7'35"

3.2 Category

Other

3.3 Channels

2

3.4 Link

https://mega.nz/file/dIFiSBZT#HQKZDLJJEhUBac_fKIki5Rhi0d7MEu6noy6xiM-LnZI

2024-ICSC (4)

Michael Gogins

Irreducible Productions
michael.gogins@gmail.com

Abstract. This piece consists of a Web page (HTML5) that embeds a GLSL shader, a WebAssembly build of Csound, a Csound orchestra, a WebAssembly build of the CsoundAC algorithmic composition system, and JavaScript code for sampling the graphics canvas used by the shader and transforming the bottom row of pixels into a chord of notes harmonised by CsoundAC and rendered by Csound. The user has interactive control over aspects of the compositional algorithm and some of the Csound instruments.

Keywords: csound, visual music, algorithmic composition, html5, webassembly

1 Program notes

This piece is implemented using the cloud-5 system for composing, performing, and publishing electroacoustic music: fixed medium, always-on or fixed duration, visual music, interactive music, and live coding. The cloud-5 system incorporates a WebAssembly build of Csound, supporting for displaying GLSL shaders, a WebAssembly build of the CsoundAC system of algorithmic composition with facilities for automating chords and scales, and the live coding system Strudel. This particular piece uses an adaption of a ShaderToy shader that is sampled to produce scales, chords, and notes rendered with Csound, and affords interactive control over aspects of both composition and rendering.

2 Biography/CV of Composer, Creator and Performers involved

I was born in 1950 in Salt Lake City, Utah, and lived there till 1973, with many trips to mountains, deserts, and unlocked university labs. My father was an inventor, my mother a fine artist and commercial artist. I have pursued poetry, photography, music performance, and music composition. I was a jazz major at the University of Utah, where I was informally introduced to electronic music by Vladimir Ussachevsky and Nyle Steiner. I have also lived in Los Angeles, New York, Seattle, and New York again. I have a B.A. in comparative religion, University of Washington, 1984.

While I was studying comparative religion, I was also studying computer music with John Rahn. Computer music gradually became my major interest. It also enabled me to make a living as a programmer, though I am now "retired" and work full-time on computer music. In the 1980s, I benefited greatly from Brad Garton's openness to non-student participation in the woof user group and concerts at the Columbia-Princeton Electronic Music Center.

I contribute code to Csound, wrote the algorithmic composition CsoundAC, maintain the Csound for Android app, host the online International Csound Users Group monthly meeting, and am on the Steering Committee of the New York City Electroacoustic Music Festival. I write articles on

computer music and create computer music. I am currently working to bring new developments in mathematical music theory into algorithmic composition software, and to create an integrated "playpen" for computer music, the cloud-5 system.

I am married to Heidi Rogers, who was owner of Frank Music Company, a classical sheet music store in New York. We live on our farm in the Catskills, and sometimes on the Upper West Side of Manhattan.

3 Technical notes

This is an online piece of electroacoustic music, rendered in your Web browser using high-resolution audio. It will play indefinitely, never ending, always changing.
The notes are played by a Csound orchestra that is embedded in this Web page using a WebAssembly build of Csound. This in turn includes the CsoundAC library for algorithmic composition, used in this piece to generate randomly selected but (I hope) musically sensible chord progressions and modulations that are applied to the generated notes.
The music is generated by sampling the bottom row of pixels from the moving image, downsampling that row into fewer pixels, and translating those pixels into musical notes from left (lowest) to right (highest). Hue is mapped to instrument, and value is mapped to loudness. Generally speaking, when a bright line moves to the bottom of the the display, you should hear some notes generated by that event.
The viewer may exercise a certain amount of control over the piece by opening the Controls. Changing the hue will change the arrangement of instruments. The tempo of both note generation and the visuals may be controlled.
When the user clicks on the Record button, the "fout" opcode is used to record the live audio to memory in the browser. When the user clicks on Pause, the recorded audio will automatically be downloaded to the user's downloads directory. Such recording may be restarted and paused again any number of times. This can be used in place of an audio loopback interface to make a soundfile from a performance.

I created the visuals for this piece by adapting Scruffy's TestShader09012024, which has an open-source license compatible with the license of this piece.
The CsoundAC library for working with chords, scales, and voice-leading implements basic ideas from Dmitri Tymoczko's work in music theory.
Code for compiling and controlling shaders is adapted from ShaderToy.com.
The algorithm for downsampling the video canvas is from Sveinn Steinarsson's MS thesis with code from https://github.com/pingec/downsample-lttb.
Csound instruments are adapted from Steven Yi (YiString and FMWaterBell), Joseph T. Kung (Kung2 and Kung4), Lee Zakian (ZakianFlute), and others.

3.1 Duration

Between 6 and 10 minutes.

3.2 Category

Live performance of visual music.

3.3 Channels

Two channels.

3.4 Link

An Ogg Vorbis recording of a sample performance of this piece is available here: https://www.dropbox.com/scl/fi/phihsji8wmwxmja2kbh27/2024-ICSC-4.ogg?rlkey=mg3ht1aspnd4a0ti10yit3uiu&dl=0.

References

1. Mandiber, Michael: The Social Media Reader. New York: NYU Press (2012).
2. Jangda, Abhinav, Bobby Powers, Emery D. Berger, and Arjun Guha. Not So Fast: Analyzing the Performance of WebAssembly vs. Native Code. https://arxiv.org/abs/1901.09056 (2019).
3. Isacson, Walter: The Innovators: How a Group of Hackers, Geniuses, and Geeks Created the Digital Revolution. New York: Simon and Schuster (2015).
4. Hafner, Katie and Matthew Lyon: Where Wizards Stay Up Late: The Origins of the Internet. New York: Simon and Schuster (1998).
5. Zuboff, Shoshona: The Age of Surveillance Capitalism. New York: Public Affairs (2019).
6. Lessig, Lawrence: Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control. City of Westminser: Penguin Books (2004).
7. Mozillla Developer Network: WebGL Reference. https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API Accessed 24 March 2024).
8. https://playcanvas.com/ (Accessed 23 March 2024).
9. Ecma: Emca International. https://ecma-international.org (Accessed 23 March 2024).
10. W3C: Making the Web work. https://www.w3.org (Accessed 23 March 2024).
11. Internet Engineering Task Force: I E T F https://www.ietf.org Accessed 23 March 2024).
12. HTML 5 Test: https://html5test.co (Accessed 23 March 2024).
13. IETF: HTTP Semantics https://www.rfc-editor.org/info/rfc9110 (June 2022).
14. W3C: Web Standards. https://www.w3.org/standards (Accessed 23 March 2024).
15. W3C: Web Audio API. https://webaudio.github.io/web-audio-api (11 March 2024).
16. Csound Developers: Csound API 6.18. https://csound.com/docs/api/index.html (Accessed 23 March 2024).
17. Gogins, Michael: Csound AC 1.0.0 https://github.com/gogins/csound-ac/blob/master/csound-ac.pdf (Accessed 23 March 2024).
18. Csound Community: The Canonical Csound Reference Manual, Version 6.18.0 https://csound.com/docs/manual/index.html (Accessed 23 March 2024).
19. Mozilla Developer Network: JavaScript Reference. https://developer.mozilla.org/en-US/docs/Web/

JavaScript/Reference (Accessed 24 March 2024).
20. Katz, Robert A.. Mastering Audio: The Art and the Science, Third Edition. Netherlands: Focal Press (2015).
21. Bassal, Dominique: The Practice of Mastering in Electroacoustics. https://cec.sonus.ca/pdf/The_Practice_of_Mastering.pdf (December 2002).
22. lazzarini, V. et al.: Csound: A Sound and Music Computing System. Springer (2016)
23. Csound Github site, http://csound.github.io.
24. Bainter, Alex: web. music. generative art. https://alexbainter.com (Accessed 23 March 2024).
25. Roos, Felix, Alex McLean, et al.: Strudel REPL. https://strudel.cc/ (Accessed 23 March 2024).
26. cloud-music: Computer Music on the Web https://AuthorA.github.io (Accessed 23 March 2024).
27. Primozic, Casey: Web Synth. https://synth.ameo.dev (Accessed 24 March 2024).
28. Roberts, Charlie: Gibber https://gibber.cc Accessed 24 March 2024).
29. Quilez, Inigo, Pol Jeremias, et al.: ShaderToy BETA https://www.shadertoy.com (Accessed 24 March 2024).

Three Chants for Computer

Fernando Egido

Independent Composer
busevin@gmail.com

Abstract. This piece experiments with the concept of intrasensory synesthesia but Instead of perceiving one sensory as another we perceive a sound feature as another one So instead of hearing colors we will perceive the time as timbre or the pitch as dynamics. To do so, I use how the perception of a musical feature affects the perception of the other musical features.

Keywords: Intrasensory Synesthesia, Parametric Music, Acousmatic Music.

1 Program notes

This piece experiments with the concept of intrasensory synesthesia but Instead of perceiving one sensory as another we perceive a sound feature as another one So instead of hearing colors we will perceive the time as timbre or the pitch as dynamics. To do so, I use how the perception of a musical feature affects the perception of the other musical features. The perception of one parameter is determined by the other ones, especially in the threshold of perception. We can achieve this using the thresholds of perception and the way that one parameter can determine the perception of another one to make parametric interdefinitions. For example, a pulse of gains of sound that is perceived as a temporal object can be converted into a timbrical object by accelerating the velocity of the pulses. beyond 16 – 20 hertz it will be perceived as no longer as a pulse but as a pitched sound. I call this a parametric morphing in which a sound object is perceived in a way and then using changing only one feature of this sound object it is perceived around different parametric centrality[3].

2 Biography/CV of Composer, Creator and Performers involved

He studied composition with José Luis de Delás at the School of Music of the University of Alcalá de Henares and received musical training in workshops with composers, analysts, and interpreters around the LIEM or the GCAC with Lachenmann, Spahlinger, Muraill, Sciarrino, Ferneyhough, Kagel, Haas, Dodge, Hidalgo, Hubert, etc... He studied Electronic Music around LIEM courses, especially with Emiliano del Cerro.
He has published several papers at international conferences and a book “Towards an Aesthetics of Cognitive-Parametric Music”.
His work “Cognitive Dissonance” was awarded at the II International Conference Sound Spaces and Audiovisual Spaces. His work “Three Chants for Computer“ was selected for the SID 2015 Conference at the New York University Steinhardt. His work “Sound Stains” was used as a component in the inauguration of the exhibition of the celebration of the 10th anniversary of the Foundation Pilar and Joan Miró on which the Miró paintings of the Reina Sofía Museum were temporally exposed. His work Parametric modulation studies was published in the magazine Quodlibet commissioned by the CDMC. In 2017 Spectropol Records included this work “Three Chants for Computer“ in a CD selection of electro-acoustic works. In 2023 MUSLAB included this work “Transmetric Variations“ in a

CD selection of electro-acoustic works.
His works have been performed at festivals and conferences such as; International CSound Conference, International Computer Music Conference (2023 Shenzhen 2024 Seoul), .abeceda Institute, MUSLAB, Ars Electronica Linz, La hora acusmática, Convergence 2022 conference in Leicester, Atemporánea Festival in Buenos Aires, Artificial Intelligence Music Creativity 2022 in Tokyo and 2020 Graz, Audio Mostly 2022 Conference in Sankt Pölten, the Sound Kitchen 2022 inside World Stage Design, Sur Aural, EVO 2021, as OUA Electroacoustic Music Festival 2020 in Osaka, International Society for Music Information Retrieval 2020 in Montreal. The Seoul International Electroacoustic Music Festival 2019, the Australasian Computer Music Conference 2019 conference in Melbourne, SID (Sound, Image, Data) 2015 conference in New York, Venice Vending Machine III, New York City Electroacoustic Music Festival (2016 – 2017- 2020-2023), JIEN in the Auditory 400 National Museum Art Center Reina Sofía, SMASH Festival, Encontres Festival in Palma Of Majorca, ACA, the Fundaçió Pilar i Joan Miró and, Nomad Roots. Please do *not* modify the page format (paper size, margins) or any of the styles included in this template. To ensure consistency in the layout, the use of direct formatting is discouraged in favor of the use of the available styles.

3 Technical notes

This work has been produced only with Csound using phase vocoding resynthesis and fof opcodes. In the first movement each sound was made with the pv opcode using phase vocoding resynthesis techniques. In the second movement, a score was created in Cubase and rendered using the VST Csound plugin using the FOF opcode each voice was produced individually. Each one of the real-time formant parameters were mapped to the p-fields via continuous control MIDI messages [4]. In the third movement, both techniques were used. The final postproduction was made in Pro-tools. You can find a detailed explanation of the work in the paper [1], [2].

3.1 Duration

11 Minutes: 15 Seconds

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

2.0 Channels

3.4 Link

https://drive.google.com/drive/folders/1S3gm804sFSIEF7lmEsjpbXmwOjT_Kxdu?usp=drive_link

References

1. Egido, F.: Intrasensory Synesthesia in Musical Composition. VI International Congress Synesthesia, Science & Art (2017)
2. Egido, F..: Towards an Aesthetics of Cognitive-Parametric Music, Lulu, New York (2011)
3. YouTube video https://youtu.be/2E89sda8JFs
4. YouTube video https://youtu.be/9JiVon85I0c

Csound Dreams in the MetaVerse (2024)

Richard Boulanger[1] and Hung Vo (aka Strong Bear)[2]

[1,2]Berklee College of Music
[1] rboulanger@berklee.edu
[2] sbear@berklee.edu

Abstract. Richard Boulanger's *Csound Dreams in the MetaVerse* (2024) uses colocation to immerse local and remote players, wearing Quest 3 XR headsets, into virtual performance spaces where they conjure and share Csound SoundObjects (orbs), that they hit, squeeze, stretch, and toss about. The 14-minute , 3-movement piece is is a structured improvisation whose score consists of a number of Boulanger's generative, immitative, procedural, percussive, modeled, textural, ambient, drone, environmental, random, rhythmic, granular, FM, AM, RM, granular, waveguide, scanned, sample-hold, and sample-based Csound instruments converted to Cabbage and imported into Unity for use with the CsoundUnity API. The sliders and triggers from the Cabbage UI have been mapped to buttons, grips, triggers, joysticks and physical hand gestures on the Quest and they appear as editable and assignable controls in the innovative and versatile *CsoundMeta* system designed by Hung Vo. In Strong Bear's Csound MetaVerse, players can see each other and collaborate with each other in the creation, modification, and performance of their Csounds and CsoundOrbs. This system and this work represent a new way to play, design, and explore unique soundworlds together, locally and remotely, in mixed reality. A unique feature of this system is how the remote players appear in the local performance space.

Keywords: Unity, CsoundUnity, CsoundMeta, Cabbage, Quest 2, Quest 3, Quest Pro, AI, AR, XR, colocation, mixed reality, passthrough

1 Program notes

Csound Dreams in the MetaVerse (2024)

Composition and Sound Design – Richard Boulanger
Unity/CsoundUnity progamming and VR system design – Hung Vo (aka Strong Bear)

Live (local) Performers – Ziaomeng (Susan) Zhong, Bethanie Liu, Miles Clark, JoNine Liu
Live (remote) Performer – Hung Vo (from Ohio, USA)
Live (local) Watcher – Richard Boulanger

Featuring the Unity and CsoundUnity programming and system design of Hung Vo (aka Strong Bear), *Csound Dreams in the MetaVerse* (2024) by Richard Boulanger is an 14-minute, 3-movement structured SoundCollage. Under the eye of the 'watcher,' whose view of the action from within a number of AI-generated VR worlds is screencast and broadcast for the audience to see and hear, as four local and one remote 'player' wearing Quest3 XR headsets, conjure *SoundOrbs* from thin air and then strike them, stretch them, twist them, toss them, catch them, share them, steal them, clone them, replace them, and eliminate them. The SoundOrbs produce a wide range of timbres and textures and serve in a number of ways to advance the narrative of the piece. Some SoundOrbs are

generative; some are explosive; some brief and momentary; some are motivic, melodic, sequential, ostinatic; some are arhythmic and others groovy. Most are synthetic, but some are sample-based. At some points in the piece, it seems like the audience is caught in the middle of a sonic food fight, whereas, at other times, they might find themselves floating in a sound cloud, or trapped in an abandoned industrial complex listening to the chaos of gasping and groaning machines; or they might find themselves gazing around a sunken underwater city listening to the singing voices of mermaids, or lost in a cave, or on the desert moon of a distant planet, or just sitting on a beach, or on a mountaintop gazing at the stars overhead listening to the music of the spheres. All of these AI-generated visual worlds compliment and reinforce the timbre, tone, temper and drama of the palettes of Csounds that each player is presented with at that point in time – when they find themselves trasported by the system to this or that location. As such, the piece is a structured improvisation in which players are presented with specific collections of SoundOrbs along their journey, each of which contains a palette (or bank) of Csounds that they can choose from, sequentially or randomly, and that they can sonically and literally reshape and transform by the movements of their hands around and through them, or by attaching 'control cables' to them and then using a variety of mapped hand gestures and button presses to more dramatically and subtly transform and modulate them. As such, the work represents a new way to compose, play, improvise, spatialize, and experience Csounds in time and shared virtual spaces.

2 Biographies

Dr. Richard Boulanger is a Professor of Electronic Production and Design at the Berklee College of Music in Boston where he has been working with some of the most talented and creative sound designers, songwriters, performers, composers, and innovators for the past 38 years.

Hung Vo (aka Strong Bear), from Vietnam, recently graduated from Berklee College of Music where he majored in Electronic Production and Design. He hold a B.E. degree in Electronics and Telecommunications from Posts and Telecommunications Institute of Technology in Ho Chi Minh City, Vietnam, and a M.S. Degree in Computer Science from Clemson University. He is the founder and CEO of *Designveloper,* a software design and development company, since 2013.

Xiaomeng (Susan) Zhong is a sound designer and audio engineer with a passion for creating immersive experiences in games and multimedia, with a B.M. in Electronic Production and Design from Berklee College of Music. Here experience includes designing custom sound effects and foley, remixing, composing, and creating interactive performative audio systems in game environments. Susan will be starting her Masters in Media Arts and Technology at UC Santa Barbara in September.

Miles Clark is a multimedia composer for games, television and film. With a BM in Game and Interactive Media Scoring at the Berklee College of Music, his work within games drives a passion for expression, oddity, and unexplored ways of storytelling through sound. He is currently working at Sparks and Shadows, a Los Angeles based screen scoring collective.

Bethanie Liu is an eighth-semester Electronic Production and Design major with minors in Electronic Production and Creative Coding at Berklee College of Music. She is originally from the United Kingdom but grew up in Hong Kong. She is a virtuoso recorder player who has won several international competitions and been featured as a soloist at Carnegie Hall in New York. She is also

an amazing Electronic Digital Instrumentalist whose virtuoso alternate-controller performances have been feature at Berklee and throughout the US. As a performer and researcher, she aspires to develop expressive real-time interfaces and tools so that people with physical disabilities can perform music. Thus, her research focuses on the development of sensor and controller systems for those with physical limitations making possible new and innovative ways for them to play music and perform with others.

JoNine (Jo9) Liu is a singer-songwriter, arranger, producer, installation artist and sound designer from Xiamen, China who fell in love with music in high school and after entering Berklee has focused on electronic music and sound design and continues to explore all fields of music and music production. JoNine worked as a Sound Designer for Tangible Future Co. in Shenzhen, China, from January 2023 to January 2024, designing over 50 expressive sound effects for LOOI Robot's various facial expressions. In addition, JoNine has served as a Producer, Composer, and Lyricist for <SEA> and <SKY> Production in Boston, MA, from 2022 to February 2023. JoNine composed and produced entire songs, designed and performed original pieces, and achieved a significant milestone by releasing music through a collaboration with SONY Music Company.

3 Technical notes

As detailed in the program note above, Csound was used for sound and effect design. Instruments were ported to Cabbage and a graphical user interface consisting of sliders, triggers, combo boxes and presets were added. Then, they were imported into Unity and are rendered, triggered, transformed, and spatialized within via the CsoundUnity API. The piece is performed live by three local players on stage and one remote player joining to play in a variety of custom-designed VR spaces over the internet. The local and remote players, and the worlds in which they are playing, are viewed by a non-performing musician who is 'watching' from within the virtual space and screencasting the action so that the audience can literally see and hear what the players are seeing, hearing, and doing in their Quest3 XR headsets as they move around on the concert stage and simultaneously explore and create CsoundScapes in the virtual worlds in which they are immersed.

3.1 Duration

The piece is 14 minutes long

3.2 Category

Immersive Audio Performance with ScreenCast Visuals and Csounds (featuring three live players on stage, and one remote player joining, appearing, and performing in the AI-generated virtual spaces with the other three over the internet via a ZeroTier VPN).

3.3 Channels

Currently Screencast from the immersive SoundWorlds in stereo (with a sub-woffer please) to a MacBook Pro from the Quest3 XR headset worn by a non-performing 'watcher'. (We are exploring

multi-channel playback solutions, and if selected for performance, might, by the time of the conference, be able to spatialize in circle 8.

3.4 Link

Public DropBox Link to several excerpts from a recent rehearsal at Berklee to give you some idea of the sounds, scenes, and roles of the performers as cast to the laptop from Quest 3 XR headset of the 'watcher'.

https://www.dropbox.com/s/oozke7cpau4ntjd/icsc2024-music-CsoundScapes_in_the_MetaVerse-boulanger.zip?dl=0

References

1. Csound site, https://csound.com/
2. CsoundQt site, https://github.com/CsoundQt/CsoundQt
3. Csound Manual site, https://csound.com/manual.html
4. Csound FLOSS Manual site, https://flossmanual.csound.com/
5. Cabbage Audio site, https://cabbageaudio.com/
6. Cabbage for Games site, https://forum.cabbageaudio.com/c/csound-for-games/10
7. Unity Download Archive site, https://unity.com/releases/editor/archive
8. MetaQuest3 site, https://www.meta.com/quest/quest-3/
9. Meta Developer site, https://developers.facebook.com/

Female Child System - Imprisonment

Anthony Di Furia

anthonydifuria.sound@gmail.com

Abstract. The composition attempts to tell an imaginary story through a "sound fable". A female child with beautiful eyes, she is incarcerated alone in a huge prison, completely dark and without windows. She is unable to speak, the only glimmer of communication is represented by the sound she hears by hitting one of the steel bars in her suspended room. Through this sound, transforming it into her mind, she embarks on a dreamlike journey; along the way, her imagination gains strength and, trying to limit it, builds a "sound mosaic" that slowly falls apart to gently lead her into a parallel reality, removing the emptiness of her perception, finally returning to her prison, keeping her life altered. She doesn't fight, she just teaches who she is. And the "sound fable" continues...
The composition is inspired by a recurring dream and is dedicated to my dear friend Ottavia.

Keywords: Csound, Synthesis, Ambisonics.

1 Program notes

The composition attempts to tell an imaginary story through a "sound fable".
A female child with beautiful eyes, she is incarcerated alone in a huge prison, completely dark and without windows.
She is unable to speak, the only glimmer of communication is represented by the sound she hears by hitting one of the steel bars in her suspended room. Through this sound, transforming it into her mind, she embarks on a dreamlike journey; along the way, her imagination gains strength and, trying to limit it, builds a "sound mosaic" that slowly falls apart to gently lead her into a parallel reality, removing the emptiness of her perception, finally returning to her prison, keeping her life altered.

She doesn't fight, she just teaches who she is. And the "sound fable" continues...

The composition is inspired by a recurring dream and is dedicated to my dear friend Ottavia.

2 Biography

He is a sound artist, software developer and sound engineer. As a composer he is interested in narrating, mixing and uniting seemingly contrasting conceptual worlds.
He studied composition at the Conservatory G.B.Pergolesi - Fermo and Electronic Music at the Conservatory G.Rossini - Pesaro under guidance Eugenio Giordani, Carmine Emanuele Cella and David Monacchi.
He worked as ambisonics spatialization assistant in the multimedia show “De Divina Proportione” by Simone Sorini and David Monacchi, as sound design in the theatrical performance “La Fuga”(Escape) in the presence of the author Gao Xingjian, Nobel prize for literature (2000) and with Eugenio Giordani he created a live electronics for the show conference "Philological and

Fantastic Bestiary" by Ermanno Cavazzoni.
His compositions have been performed in international conferences and festivals such as FKL soundscape meeting (Florence, Italy 2014), Linux Audio Conference (ZKM in Karlsruhe, Germany 2014), La Chambre Blanche (Ville du Quebec, Canada 2014), TeverEterno (Rome, Italy), Pianpicollo Selvatico (Levice, Italy 2016), Csound30 (Maynooth University, Ireland 2016), sfsound (San Francisco, USA 2019), ICSC2019 (Cagli, Italy 2019), ICSC2022, ISAC-2023 Sonosfera. He worked as sound designer on the film "Dusk Chorus, based on fragments of extinction by David Monacchi", directed by Nika Saravanja and Alessandro D'Emilia winner of several awards at film festivals international.
In 2014 at the Chambre Blanche (Quebec) he created a multimedia installation in ambisonics called "Beyond the human atom". Since 2017 he is an Apple software developer for his own applications.
From 2018 to 2020 he worked as a multimedia software developer and software interconnection for the project Fragments of Extinction and Sonosfera Pesaro by David Monacchi. In 2019 together with Eugenio Giordani, Alessandro Petrolati, Laura Muncacio and Enrico Francioni he organizes ICSC2019 Csound International Conference.

3 Technical notes

The composition is made entirely in Csound. The piece was made only with synthetic sounds, starting from the sound simulation of a steel bar.
Implemented algorithms:
1 - Steel bar synthesizer through additive synthesis and its manipulation.
2 - UDO Ambisonics 6th order encoder.

3.1 Duration

7'09”

3.2 Category

Electroacoustic music on fixed medium (Ambisonics)

3.3 Channels

21.2

3.4 Link

https://drive.google.com/drive/folders/1Mrq6rMg7Omf7NZzRawCJ63XSNOWBtEaE?usp=share_link

Ordinary Rehearsals

Antonio Scarcia

Liceo "De Ruggieri" Massafra Italy
ant.scarcia@gmail.com

Abstract. "Ordinary Rehearsals" is an electroacoustic piece that utilizes digital techniques inspired by the traditional workflows of tape studio recording. The piece utilizes Csound for sound synthesis through sampling, articulating complex sound gestures from initially contrasting materials. These materials are designed to evolve into a dialogue, seeking moments of equilibrium. Scores are algorithmically generated within a computer algebra system, ensuring a sophisticated integration of computational precision with artistic expression. This piece intricately explores the tension and dialogue between disparate sound elements.

Keywords: Generative, Sampling, Csound.

1 Program notes

"Ordinary Rehearsals" for digital media simultaneously explores and reflects on the relationship between materials and musical gestures. Initially, non-pitched and tonic materials are introduced as antagonistic elements, but they eventually merge, achieving sinergically a stable state of tension. The piece creates a virtual soundscape, incorporating indoor audio recordings and tuned string instrument sounds, all manipulated within the Csound environment. Scores were generated using a general-purpose numerical environment with a generative approach. This piece was performed with an honorary mention at the Musica Nova International Electroacoustic Music Competition in Prague, 2013.

2 Biography/CV of Composer, Creator and Performers involved

Antonio Scarcia (1959) earned his degree in Electronic Engineering from the University of Padua and holds a postgraduate diploma in Signal Processing from the University of Bari. He also received an academic diploma in Electronic Music with Honors from the Conservatory of Bari, studying under the supervision of Francesco Scagliola. He has held various teaching positions as an adjunct professor at the Genoa Conservatory of Music from 2011 to 2021 and served as the professor of Electroacoustic Music Composition at the Conservatory of Salerno from 2022 to 2023. His works for digital media have been featured at major events, including various editions of the NYCEMF in 2022, 2021, and 2019; ICMC in 2014, 2013, 2012, 2010, and 2007; the North Carolina Computer Music Festival in 2008; SMC in 2012, 2010, and 2009; the Mantis Festival in 2010; CIM in 2018, 2016, 2014, 2012, and 2010; EMuFest in 2013, 2012, 2011, and 2010; SICMF in 2013; ICSC in 2013 and 2022; and the Musica Nova Competition, where he received honorary mentions in 2016 and 2013, and first prize in 2011.

3 Technical notes

"Ordinary Rehearsals" employs digital techniques reminiscent of classic tape studio methods, involving a typical step-by-step process. Initially, all scores are generated using a computer algebra environment with a generative approach. Alongside a collection of original recordings, the composition unfolds through a series of sound gestures, crafted using Csound's basic sampling techniques in deferred time. Extensive post-processing is conducted in both the time and frequency domains of Csound renderings. It is important to note how a very basic approach in the construction of the orchestra, combined with complex scores generated algorithmically, allows Csound to create interesting sound textures.

3.1 Duration

6:00

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

Original format is stereophonic (2); the work is ideally intended for multichannel projection through the author's direction at mixer console.

3.4 Link

https://drive.google.com/file/d/1AObIcF8Fs8Ak4btFugMkcPUOVJWN9buj/view?usp=sharing

WS Gluing Map

Juan J.G. Escudero

jjgemplubg@gmail.com

Abstract. This piece explores certain connections between music and geometry.

Keywords: music and mathematics; spectra of aperiodic temporal sequences; electronic music on fixed medium

1 Program notes

From a formal point of view this work is based on a combinatorial description of the Seifert-Weber space, which is a multiconnected hyperbolic three-manifold where the faces of a dodecahedron are identified after some rotations. The construction of a random simplicial complex of the three-manifold originates from a starting triangulation or axiom. FM synthesis, plucked strings and other Csound instruments are used. The function tables are obtained from spectra of time quasicrystals and certain models of multiperiodic variable stars light curves based on analogous temporal structuring.
Multiconnected manifolds are candidates for the spatial structure of the Universe. One of the consequences would be the observation of the same part of the cosmos in different places of the sky and it appears due to the presence of a closed loop in the manifold. Some musical correspondences of this facts are explored.

2 Biography/CV of Composer, Creator and Performers involved

Juan J.G. Escudero is a composer and researcher based in Madrid (Spain). He received his musical education at several centres and conservatoires and studied composition with Francisco Guerrero Marín in Madrid. He has carried out research and teaching activities in mathematics, physics and music technology at various universities. The results of his studies in the fields of algebra, geometry and astronomy -published in scholarly journals and books- have been some of the main guides to formalization procedures. Harmonizations of aperiodic ordered temporal sequences, which are on the basis of the formal and rhythmic structures play a major role in several of his instrumental and acousmatic works. More recent formal approaches are related with the analysis of the topological invariants of aperiodic tiling spaces and the construction of singular hypersurfaces in algebraic geometry. Extramusical influences are connected mainly with philosophy, poetry and visual arts. Monographic albums have appeared on Neuma Records (USA) and Sargasso (UK). His musical works are published by Universal Edition.

Juan J.G. Escudero

Technical notes

2.1 Duration

7:46

2.2 Category

Electroacoustic music on fixed medium

2.3 Channels

2 or 8 channels

2.4 Link

https://drive.google.com/file/d/1NNw5gmeXm8JFh-mC4QeG9hmp3vvl_fLl/view?usp=sharing

Ripples in the Fabric of Space-Time

Jon Christopher Nelson

University of North Texas College of Music
jon.nelson@unt.edu

Abstract. *Ripples in the Fabric of Space-Time* constitutes the fourth movement of Jon Christopher Nelson's *The Persistence of Time and Memory*. The work was composed making extensive use of Csound physical modeling opcodes driven by MPE data from a roli seaboard controller. The work explores correlations between modulated physical sound models and sampled sound environments.

Keywords: Csound, MPE, roli, physical modeling

1 Program notes

Ripples in the Fabric of Space-Time imagines a sound world filled with the "chirps" that result from two black holes colliding. As black holes collapse into one another they create a highly deformed new black hole that emits gravitational waves from its equator. These gravitational waves move up and down in frequency a few times before they die, creating "chirps." In this work aural chirps disrupt our temporal expectations, resulting in an animated soundscape filled with rapid and playful transformations between allusions to acoustic instruments, sonic environments, and percussive noises. This composition represents the fourth movement of Nelson's six-movement acousmatic odyssey, *The Persistence of Time and Memory.*

2 Biography/CV of Composer, Creator and Performers involved

Jon Christopher Nelson (b. 1960) is a Professor of Composition at the University of North Texas College of Music where he is as an associate of CEMI (the Center for Experimental Music and Intermedia). Nelson is perhaps best known for his work in computer and electronic music. His electroacoustic compositions have been performed widely at festivals and conferences throughout the United States, Europe, Asia, and Latin America. He has been honored with numerous awards including fellowships from the Guggenheim Foundation, the National Endowment for the Arts, and the Fulbright Commission. He is the recipient of Luigi Russolo Prize (1995), Bourges Prizes (1996, 1997, 1999, 2002 and the Bourges Euphonies d'Or prize in 2004) and the International Computer Music Association's Americas Regional Award (2012) and Music Award (2020). He has composed in residence at Sweden's national Electronic Music Studios, the Visby International Composers Center and at IMEB in Bourges, France.

3 Technical notes

The sound materials for this composition were generated making extensive use of Csound's physical modeling opcodes within the Cabbage framework. These opcodes were controlled within an MPE

context using a roli seaboard to provide control data for the opcodes in the generation of synthesized audio samples. The composition also includes some field recordings that complement the synthetic sounds. All effects used are coded in Csound, including extensive use of Nelson's waveguide mesh reverb, Mverb. All audio generation and manipulation exclusively uses Csound mixed with some field recordings using Reaper (no plugins).

3.1 Duration

7:50"

3.2 Category

Electroacoustic music on fixed medium

3.3 Channels

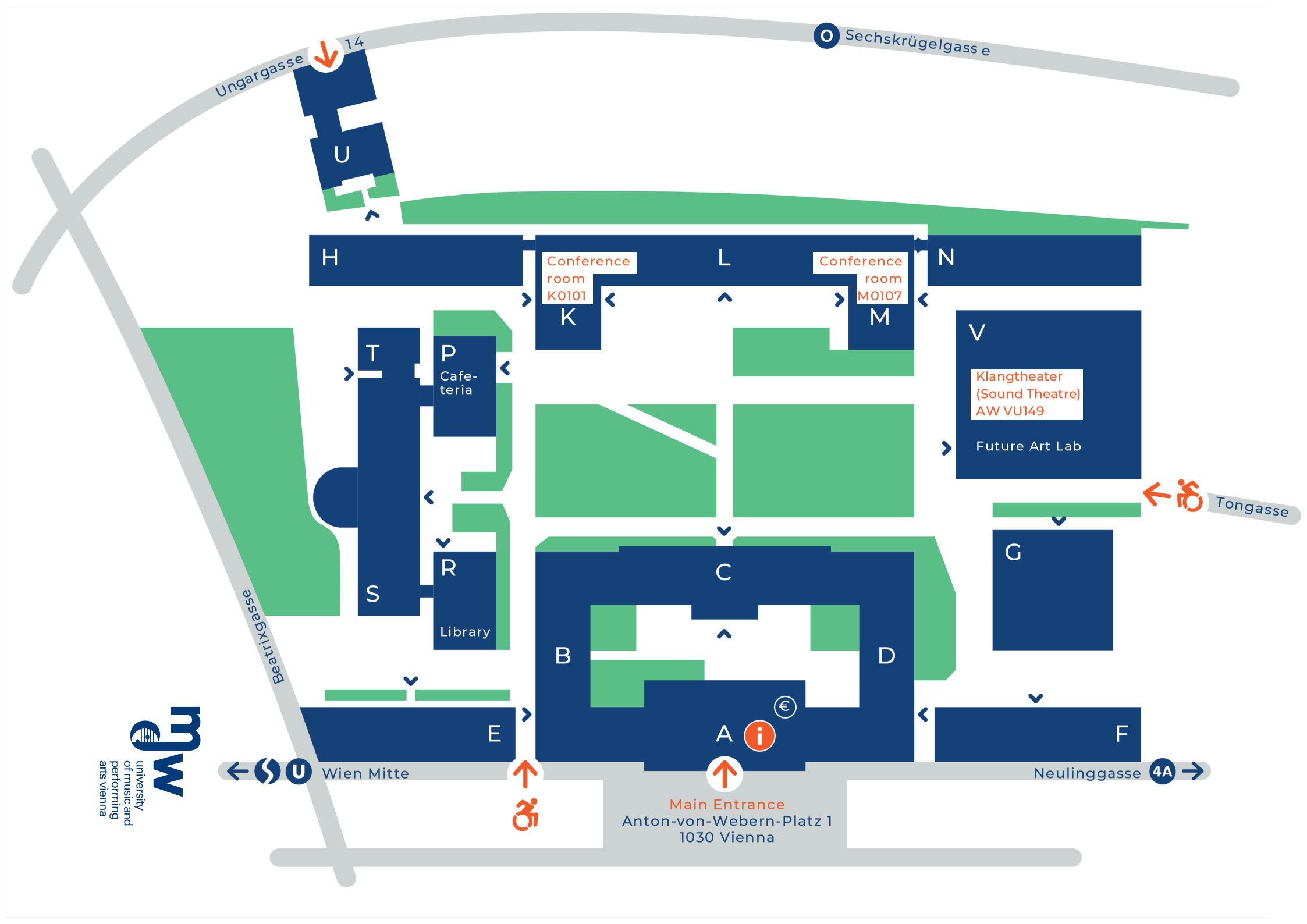
The composition is stereo fixed media, but I would prefer to diffuse the composition over a 16-channel (or greater) system

3.4 Link

https://drive.google.com/file/d/1Xwpk10Tnb73z1vIBz2IvmQChrdz9kJff/view?usp=sharing

LIST OF CONFERENCE CONTRIBUTORS:

Abedian, Arsalan	Hannover University of Music, Drama and Media (HMTMH), Germany
Annese, Daniele Giuseppe	Conservatorio "Niccolò Piccinni" di Bari, Italy
Ballerini, Lorenzo	Conservatory of Trapani, Italy
Bear, Strong	Berklee College of Music, United States
Boulanger, Richard	Berklee College of Music, United States
Brandtsegg, Øyvind	Norwegian University of Science and Technology – NTNU, Norway
Byrne, Shane	Technological University of The Shannon (TUS), Ireland
Carty, Brian	Institute of Art, Design and Technology, Dún Laoghaire, Ireland
Della Vittoria, Gianni	Liceo Artistico e Musicale "A. Canova" di Forlì, Italy
Di Furia, Anthony	Conservatorio "Niccolò Piccinni" di Bari, Italy
Di Liscia, Oscar Pablo	Universidad Nacional de Quilmes, Argentina
Doati, Roberto	Independent, Italy
Dunne, Patrick	Technological University of the Shannon, Ireland
Egido, Fernando	Independent, Spain
Ernandez, Giuseppe	Conservatory of Trapani, Italy
Escudero, Juan	Independent, Spain
Ferguson, Mark	Independent, United Kingdom
Ffitch, John	University of Bath, United Kingdom
Gogins, Michael	Irreducible Productions, United States
Greco D'Alceo, Jacopo	Independent, France
Grund, Tim-Tarek	mdw – University of Music and Performing Arts Vienna, Austria
Han, Jinhao	Sichuan Conservatory of Music, China
Heintz, Joachim	Hannover University of Music, Drama and Media (HMTMH), Germany
Hofmann, Alex	mdw – University of Music and Performing Arts Vienna, Austria
Hofmann, Jan Jacob	Independent, Germany
Izadyar, Parham	Independent, Iran
Jagwani, Aman	Maynooth University, Ireland
Janevska, Marijana	Hannover University of Music, Drama and Media (HMTMH), Germany
Johannes, Tarmo	Independent, Estonia
Khoshsabk, Amin	Independent, Iran
Kim, Seokyeong	Maynooth University, Ireland
Kobayashi, Ken	Berklee College of Music, United States
Lazzarini, Victor	Maynooth University, Ireland
Liu, Bethanie	Berklee College of Music, United States
Madrenys Planas, Albert	Maynooth University, Ireland
McDonnell, Thom	Sound Training College, Temple Bar, Ireland
Moqanaki, Ghazale	Independent, Iran
Nelson, Jon Christopher	University of North Texas College of Music, United States
Pelleboer, Hans	Perceptual Engineering, Netherlands
Reina, Massimo	Conservatory of Trapani, Italy
Scagliola, Francesco	Conservatorio "Niccolò Piccinni" di Bari, Italy
Scarcia, Antonio	Liceo De Ruggieri Massafra, Italy
Sevilgen, Serkan	Istanbul Technical University (MIAM), Turkey
Silvi, Giuseppe	Conservatorio "Niccolò Piccinni" di Bari, Italy
Sosa, Jean-Basile	Independent, France
Speicher, Leon	Hannover University of Music, Drama and Media (HMTMH), Germany
Tremblay, Pierre-Alexandre	Independent, England
Vitucci, Francesco	Conservatorio "Niccolò Piccinni" di Bari, Italy
Von Reusner, Clemens	Independent, Germany
Walsh, Rory	Dundalk Institute of Technology, Ireland
Yi, Steven	Independent, United States
Zhong, Xiaomeng	Berklee College of Music and UC Santa Barbara, United States


Ungargasse
14
Sechskrügelgasse
U
H
Conference room K0101
K
L
Conference room M0107
M
N
T
P
Cafe-teria
S
R
Library
V
Klangtheater (Sound Theatre) AW VU149
Future Art Lab
Tongasse
G
C
B
D
A
E
F
Beatrixgasse
university of music and performing arts vienna
Wien Mitte
Neulinggasse
4A
Main Entrance
Anton-von-Webern-Platz 1
1030 Vienna