Ways Ahead

Ways Ahead:
Proceedings of the First International Csound Conference

Edited by

Joachim Heintz, Alex Hofmann
and Iain McCurdy

CAMBRIDGE
SCHOLARS
PUBLISHING

Ways Ahead: Proceedings of the First International Csound Conference,
Edited by Joachim Heintz, Alex Hofmann and Iain McCurdy

This book first published 2013

Cambridge Scholars Publishing

12 Back Chapman Street, Newcastle upon Tyne, NE6 2XX, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN (10): 1-4438-4758-5, ISBN (13): 978-1-4438-4758-2

TABLE OF CONTENTS

Foreword ix

History

Csound Past, Present and Future 2
Richard Boulanger

How to become a Csound Root? 9
Interview with John ffitch

Licensing Csound to LGPL 13
Interview with Richard Boulanger

Development

User-Developer Round Table I: The Technology of Csound 24
Victor Lazzarini

Writing Csound Opcodes in Lua 32
Michael Gogins

The Csound API 43
Interview with Michael Gogins

Csound and Object-Orientation: Designing and Implementing Cellular Automata for Algorithmic Composition 48
Reza Payami

The Hadron Plugin and the Partikkel Opcode 57
Interview with Øyvind Brandtsegg

Developing Csound Plugins with Cabbage 64
Rory Walsh

Wavelets in Csound 83
Gleb G. Rogozinsky

Music

Performing with Csound: Live and with CsoundForLive 96
An Interview with Dr. Richard Boulanger (a.k.a. Dr. B.)

Fingers in the Waves 109
An Interview with John Clements

Spinning Pendulum Clock 112
An Interview with Takahiko Tsuchiya

Razor Chopper 114
An Interview with Øyvind Brandtsegg

..*Ed Io Sto Al Centro* 119
An Interview with Enrico Francioni

Geographies 122
An Interview with Giacomo Grassi

Zeitformation 125
An Interview with Jan Jacob Hofmann

...und lächelnd ihr Übel umarmen... 130
An Interview with Wolfgang Motz

The Echo 133
An Interview with Reza Payami

Três Quadros Sobre Pedra 136
An Interview with Luís Antunes Pena

Vineta 139
An Interview with Elke Swoboda

Chebychev 141
An Interview with Tarmo Johannes:

The Rite of Judgment 144
An Interview with Nicola Monopoli

Csound Haiku: A Sound Installation 146
Iain McCurdy

Continuous Flow Machines 149
Clemens von Reusner

Usage

The Csound Journal 156
Interview with Steven Yi and James Hearon

Developing CsoundQt 159
Interview with Andrés Cabrera

Python Scripting in CsoundQt 163
Andrés Cabrera

Composing Circumspectral Sounds 170
Peiman Khosravi

Creating Reverb Effects using Granular Synthesis 181
Kim Ervik and Øyvind Brandtsegg

Introducing Csound for Live 188
Dr. Richard Boulanger

Composing with Blue 203
Steven Yi

The UDO Database 215
Steven Yi and Joachim Heintz

Education

User-Developer Round Table II: Learning and Teaching Csound 220
Iain McCurdy

A Personal View on Teaching Csound 229
Gleb G. Rogozinsky

PWGL: A Score Editor for CSound 243
Massimo Avantaggiato

QUINCE: A Modular Approach to Music Editing 257
Maximilian Marcoll

Csound at HMTM Hannover 266
An interview with Joachim Heintz and Alex Hofmann

The Csound Real-time Collection 270
Iain McCurdy

Teaching with Csound 276
Interview with Peiman Khosravi and Rory Walsh

Collaborative Documentation of Open Source Music Software:
The Csound Floss Manual 282
Alex Hofmann, Iain McCurdy, and Joachim Heintz

Impressions of the First International Csound Conference 289

Schedule of the First International Csound Conference 294

Editors 299

FOREWORD

The First International Csound Conference, held at the Hanover University of Music, Drama and Media (HMTMH) between the 30th September and the 2nd October 2011, marked the first time that the principle people involved with Csound—in existence since 1986—met in person. Contacts and relationships had long been established through discussion lists and publications, but meeting together in one location felt like the Csound community had drawn closer together—it finally felt like a real community.

I. The Conference

It is surprising to note that it took 25 years for the first Csound conference to come about, but sometimes an unexpected event can be fortuitous, someone simply asking the right question at the right time. It was whilst involved in a research project at the HMTMH that Alex Hofmann, who was at the time more familiar with MaxMSP and SuperCollider, listed what he regarded as the pros and cons of Csound in its current state. Based on his conclusions, one of the proposals he made was: a Csound conference!

When this idea was pitched towards the Csound community it prompted many enthusiastic reactions. With the kind support of the HMTMH, Joachim Heintz, head of the electronic studio in the Institute for New Music *Incontri*, was able to host this conference. It is a measure of the conviction and dedication of many within the Csound community that so many travelled so far in order to attend. The conference welcomed people from Ireland, UK, Norway, France, Italy, Switzerland, Germany and from as far afield as Estonia, Russia, Iran, USA and Japan.

II. …And Its Proceedings

Some documentation pertaining to the conference already exists at www.cs-conf.de. This was the original website for the conference and it contains an outline of the events that took place, papers presented, workshop overviews and concert programmes. Videos from round tables,

paper sessions and workshops are available at www.youtube.com/user/csconf2011 but it is thanks to the initiative of Cambridge Scholars Publishing that the proceedings of the conference are now published as a book. The three authors of this foreword had already worked closely together on The Csound FLOSS Manual so it felt appropriate to share this new challenge together.

This *was* a real challenge because this book is not simply a documentation of the conference. It was our publisher who encouraged us to take the opportunity to work on a book that contained a lot of new material. Of course, the conference papers and workshop descriptions are all represented—and in many cases updated and expanded—but the reader will find a lot more.

III. Content of this Book

The first part is dedicated to Csound's history, a history that goes back to the very origins of computer music. This publication opens with Csound legend Richard Boulanger's original conference keynote, *Csound Past, Present and Future,* which is followed by an interview with Csound's chief maintainer John ffitch, *How to Become a Csound Root.* This part of the book concludes with an interview with Richard Boulanger in which he discusses the process and reason for moving Csound's licence to LGPL.

The second part of the book showcases various aspects of Csound's development. Victor Lazzarini's article, *The Technology of Csound,* is not just a summary of the conference's first user-developer round table discussion, it is an in-depth "snapshot 2012" of Csound's internals, from the implementation of the new parser to the porting of Csound to mobile devices. There is a new interview with Michael Gogins about The Csound API as well as his article about writing Csound opcodes in Lua, another of his innovations as a Csound core developer. Reza Payami's discussion about Csound and object-orientation documents a paper he presented at the conference while the last three texts in this chapter are completely new: Partikkel Audio's Øyvind Brandtsegg is interviewed about his Csound powered granular synthesis plug-in "Hadron". Rory Walsh has written a comprehensive explanation of his front-end for Csound, "Cabbage", which offers new ways of exporting Csound instruments as plug-ins. Gleb Rogozinsky describes his implementation of a new Csound GEN routine in *Wavelets in Csound.*

Csound is principally used for the creation of music and accordingly the central part of this book is dedicated to the music featured at the conference. Not every piece that was performed at the conference could be

featured in this book but the reader will find interviews with many of the contributing composers, which will provide an insight into the myriad of ways in which Csound can be used as a compositional tool.

The fourth part, *Usage*, showcases different ways of using Csound. *The Csound Journal*, edited by James Hearon and Steven Yi, can be considered the major platform for users to present and discuss different aspects of Csound usage, so an interview with both editors seemed timely here. For most Csound users, Andrés Cabrera's "CsoundQt" front-end is their main way of interacting with Csound, so both an interview with him about this project and his original conference contribution about Python scripting in CsoundQt are presented here. Peiman Khosravi describes a method of using Csound's famous streaming phase vocoder opcodes for multichannel work in his article, *Composing Circumspectral Sounds*. This is a new and extended version of the original conference paper, as is Kim Ervik and Øyvind Brandtsegg's contribution, *Creating Reverb Effects Using Granular Synthesis.* Two more of the conference's workshops appear also in a new light in this part: Steven Yi introduces his own Csound front-end in *Composing with Blue*, providing a thorough explanation of design and usage, and Richard Boulanger explains *CsoundForLive*, which connects Csound with two of the most extensively used audio applications: Ableton Live and MaxMSP. The ever-expanding User Defined Opcodes Database, established and maintained by Steven Yi, is the focus of the last interview in this part of the book with Steven and Joachim Heintz.

Because of its history and its simple signal flow language, Csound is well known as a tool for teaching audio programming and digital sound processing. For this reason, the final part provides an overview of Csound within various aspects of education. Iain McCurdy contributes a summary of the second round table discussion, which used the theme: *Learning and Teaching Csound.* Gleb Rogozinsky describes his experiences of teaching Csound in Saint-Petersburg State University of Film and Television in *A Personal View on Teaching Csound*. Massimo Avantaggiato outlines his use of the programming language PWGL for composition in *PWGL: a Score Editor for Csound*. Maximilan Marcoll describes his program "quince" in the chapter: *A Modular Approach to Music Editing.* Alex Hofmann and Joachim Heintz talk about the role of Csound at the HMTM Hannover from the perspectives of former student and lecturer respectively. Iain McCurdy introduces his popular collection of real-time Csound examples in another chapter and interviews Peiman Khosravi and Rory Walsh about their experiences in teaching with Csound in England and Ireland. At the end of this final part of the book, the three editors

introduce “The Csound FLOSS Manual”, Csound’s collaboratively written textbook.

IV. Ways Ahead

We hope that the publication of these articles and interviews will suitably mark the momentous occasion that was the first Csound Conference and afford people, who were unable to attend the conference in person, the opportunity to benefit from the wealth of intellect, experience and talent that was present during those three days. We believe that this book is also a documentation of the many different activities within the Csound community—developers and users, musicians and programmers, students and teachers—at this time, 25 years after its creation by Barry Vercoe at MIT.

Csound has come a long way from demanding to be invoked from a terminal window. Now users have the choice of a variety of front-ends, such as *CsoundQt*, *WinXsound*, *Blue* and *Cabbage*, with which to interact with Csound. Even further abstraction is evidenced in the seamless use of Csound within *PD*, *MaxMSP*, *Ableton Live* and other software via its API or as a plug-in.

It seems that Csound's forward thinking design has facilitated a wide diversity of musical applications. Whilst most music software focusses on one or two applications and styles, Csound is equally adept at producing Electroacoustic music, techno, imitations of conventional musical instruments, live electronics, sound installations, self-generating musical processes... the list goes on.

Csound defies its age through the living work of musicians, researchers and teachers. We hope that this book can contribute to these “Ways Ahead”, as the conference certainly has done.

We would like to thank all the contributors for their efforts and their willingness to publish their thoughts here. Thanks are also due to Anna Heintz-Buschart for her help in completing the manuscript, and to Carol Koulikourdi and Amanda Millar from Cambridge Scholars Publishing for their patience and kindness.

Hanover, Belfast, Vienna. January 2013.
Joachim Heintz, Iain McCurdy, Alex Hofmann.

HISTORY

CSOUND PAST, PRESENT AND FUTURE: KEYNOTE FOR THE OPENING OF THE FIRST INTERNATIONAL CSOUND CONFERENCE

RICHARD BOULANGER

It is a great honor to have been asked to present the keynote at this, the *First* International Csound Conference.

Thank you so much to Joachim Heintz and Alex Hofmann for inviting me, for inviting all of us, and thanks to the Hanover University of Music, Drama and Media for supporting them and helping them to host this historic gathering.... a gathering that points to the future—The Ways Ahead—for Csounders and for Csound.

I am so humbled to be here in the company of so many brilliant Csounders from all around the world. For years now, we have known each other through our email, our code, our music, but to finally meet in person—face to face—is absolutely incredible. I will start my greeting with extreme gratitude. Your work, your research, your questions, your feedback, your suggestions, and most importantly your sounds, your instruments and your music have so inspired me, have so enlightened me, and have given me so much to share with my students and to explore in my music. Truly, your work, your questions, your suggestions, and your *sometimes* frustrations - all served to guide and propel Csound into the future - making it *always* current and always relevant - making it the "go-to" synth.

To those developers here among us - John ffitch, Michael Gogins, Victor Lazzarini, Andrés Cabrera, and Steven Yi, I offer my deepest gratitude. You kept Csound alive; and as a result, you kept our musical creations, our dreams... alive.

Thanks to you, Csound is "Future-Proof". New hardware comes along. New operating systems come along. New protocols. New sensors and interfaces. And you have made sure that Csound keeps on "rendering" the classics while taking advantage of all of these new possibilities.

The "prime directive" for Csound has been "100% backward compatibility". We will not allow new technology to trash yesterday's music. We will not throw away the music when we upgrade the machinery—the baby with the bath water. And thanks to you, this is truly the case. This is a unique quality of Csound. It still works. It *all* still works! The earliest pieces. The first tutorials. They all still run. As we all know, Csound carries on the tradition of the Music-N languages developed by Max Mathews at Bell Labs in 1956/57. And, in Csound much of his code and examples still work too! With Csound, we have all been able to stand on the shoulders of giants and learn from the masters who have come before us. This is possible because of the selfless dedication of these amazing developers.

It all started at MIT where Barry Vercoe wrote the earliest versions of Csound. They were based on the *MusicV* language of Max Mathews as manifested first in his *Music360* language (for the IBM360 mainframe), and then in his *music11* language (for the PDP11 MiniComputer). Adding opcodes and functionality, composing original works, hosting conferences, offering workshops, commissioning composers, and delivering papers, Barry Vercoe carried the Csound torch from 1986-1996. (It could be argued that Csound was born in 1977, and called *music11* after the PDP11 computer on which it ran, and was then rewritten in the new "portable" C Programming Language in 1986 with an appropriate name change to Csound). In fact, I composed Trapped in Convert (in *music11*) at Barry's 1979 summer workshop and, in 1986, came back to MIT after my Ph.D. (in Cmusic ironically) to work again with Barry as he was moving from the PDP11 to the VAX 11780. During that period, we were revising and using "Trapped in Convert" as the main test suite for his new port of *music11* - Csound.

In 1995 or so, Barry went off to develop a proprietary version of Csound for Analog Devices and passed the torch for "public Csound" to John ffitch and myself. (Actually John and I were there working with him at Analog Devices too; and it was exciting to have Csound running on the ADI SHARC DSP (we called this version SharCsound) and, for that time period, Csound was doing unheard-of real-time, multi-channel signal processing and synthesis! (In fact, Barry added some amazing opcodes to SharCsound, which, sadly, have never made it into "public Csound" as they are the proprietary property of Analog Devices.)

From the fall of 1989, John ffitch had been working on PC, Linux, and Mac versions of Csound and he and I were communicating quite regularly over the Internet. But it was the Analog Devices project that brought us to work together in the same lab; sent us off to deliver papers and demos at

conferences together; and got us "performing" together - playing Csound live! (Something that we will do again here at this conference in tonight's concert.)

Barry always said that passing Csound on to John gave him the time to follow his muse. And thanks to John's indefatigable work on Csound, we were all able to follow our muse. Because of John's work, we could use Csound on virtually any and every computer that we could get our hands on. John ffitch's work on Csound opened the doors of the MIT Media Lab to students, composers and musicians around the world. The Csound mailing list, which he started and managed, was our way to communicate, with him, and with each other. From its very inception, this has been a respectful and insightful dialog; a rare exchange between musicians, researchers, scholars, students—experts and beginners—of all ages, and from the broadest range of technical and musical experience, aesthetic, and perspectives. Some of us liked MIDI, some of us liked note-lists (OK, that's pushing it—none of us liked note-lists), some of us liked Techno, some of us liked Disco, some of us liked Xenakis (Xenakis and Disco in the same sentence - pretty good right!). All kidding aside, this rich conversation continues to this day, and it serves as a model to all creative online communities. In the Csound community, which John ffitch so fundamentally helped to build—everyone is welcome, everyone is heard, everyone's questions are answered, and everyone's suggestions and ideas seem to get implemented!

I have been working with Csound for more than 30 years now, and I am still discovering new things, new sounds, new ways of working, new ways of thinking, and new ways of performing and interacting. It is thanks to the developers here, and to those who could not make it, but whose contributions are sited in the Csound manual, that we can all continue to grow with Csound. In particular, I would like to extend a special thanks to: John ffitch for Csound, Csound5, Csound6, and WinSound; to Gabriel Maldonado for CsoundAV; to Matt Ingalls for MacCsound and csound~; to Davis Pyon for csound~; to Øyvind Brandtsegg for Hadron; to Peiman Khosravi for FFTools; and to Jinku Kim, Enrico De Trizio, and Colman O'Reilly for CsoundForLive; to Richard Dobson for CDP (and all the streaming vocoder opcodes); to Victor Lazzarini for the Csound API; to Michael Goggins for CsoundAC and Silence; to Steven Yi for Blue; to Jean Piché and Olivier Bélanger for Cecilia and the OLPCsound apps TamTam and SynthBuilder; to Rory Walsh for Lettuce; to Stefano Bonetti for WinXound; and especially to Andrés Cabrera for his award-winning QuteCsound (now called CsoundQt). Their intuitive, brilliant and powerful "front-ends" to Csound, with all sorts of built-in "features", "short cuts",

and "assets" (including GUI tools, Multi-channel output and support for all sorts of hardware, expanded MIDI and OSC, Python, and so much more), have made Csound a tool for education, composition, research, concert performance, and commercial production, and as such, have helped to introduce Csound to a wider audience of creative musicians—have brought Csound from the "media lab" to the independent media artists around the world.

In addition to the inspiring front-ends that have been developed by these fantastic Csounders, the work on the Csound manual has been incredibly important to the longevity of the program and to the education of the community. It was because of the fact that Barry's original *music11* manual was so terse and difficult to understand that I was inspired to write The Csound Book. I had read it dozens of times, from cover to cover; filling every margin with notes and comments, but there was so much that I still did not get. Sure, it was heralded (by Barry at least) for its "concise technical descriptions" of *all* the *music11* opcodes; but the language, and the code fragments were completely alien (to me at least, and I suspected that they would not totally "click" for many other composers and practicing electronic musicians). And so, in an indirect, but quite literal way, it was Barry Vercoe's *music11* and subsequent Csound manual that inspired me to write "The Csound Book" (for the rest of us!), which explained how things worked and, as a rule, contained *complete* working instruments in every tutorial chapter. According to Barry, the *music11* and Csound Manuals were "reminders", for experts, about the working parameters for the opcodes, "not tutorials" on how the opcodes actually worked—to know that, "read the code". Well, electronic musicians and composers (like me!) needed more. And so, I insisted that "The Csound Book"... WORKED! Through it, we learn by hearing, touching, tweaking, revising, remixing and composing. In fact, as I was getting started writing it, and I had already signed the contract with MIT Press, I quickly realized that I didn't actually know enough to write such a comprehensive text on my own, and so, I reached out to the Csound community and invited the subject specialists to write working tutorial chapters on each of their specific areas of expertise. The "Csound Book" teaches computer music, software synthesis, and digital signal processing - through Csound, and the best computer music professors from around the world are on the faculty. I am truly grateful to each and every one of them, and I think that you might be too!

The Csound Book took five years to write and was published in 2000. Since then, things have changed in the world of Csound and especially in the Csound Manual department. The current Csound Manual (over 1500

opcodes covered in over 3000 pages!) has come a long way with the help of some dedicated Csounders - Jean Piché, David M. Boothe, Kevin Conder, John ffitch, Michael Gogins, Steven Yi, and many others who labored to get the words right, but most importantly, through the efforts of Menno Knevel, who has invested years into making sure that *every* opcode has a working *musical* example - he strove to get the *sounds* right. The addition of his musical models to the manual, and those that he collected, edited, and curated, has been huge. Thank you to everyone who has contributed to this important document, and to those who continue to breathe life into it with new words, new instruments, and new opcodes every day.

But wait, there is another. Another Csound manual (as if 3000 pages is not enough)! In fact it is not. And to coincide with this First International Csound Conference, a team of three - Joachim Heintz, Alex Hofmann, and Iain McCurdy are releasing a new, collaboratively written, Csound Manual/Book - called the Csound FLOSS Manual (It's really the 2010 Csound Book but they were being polite). It is amazing and incredibly inspiring. There have been so many new things added to Csound over the years, and there are so many powerful new ways of working with Csound. This book, filled with working *instruments* and examples, connects yesterday's Csound with tomorrow's Csound - TODAY! It is a "must read" for every Csounder.

It turns out, in fact, that these are not the only two Csound books by any means. There have been a number of significant texts written on the language and they all bring something special to our understanding and appreciation of it. Riccardo Bianchini and Alessandro Cipriani wrote "Virtual Sound" around the same time as The Csound Book; Andrew Horner and Lydia Ayers wrote "Cooking with Csound" a couple years later. Jim Aikin has just released "Csound Power"; And Giorgio Zucco just released Digital Synthesis: A Practical Laboratory in Csound.

Between the years that separated the publication of these fantastic books, there have been two online magazines, distributed quarterly, whose articles and instruments have helped beginner and expert alike share their work, introduce unique approaches, and discover new algorithmic, compositional, and production possibilities. Starting in around 2000, Hans Mikelson pretty much single-handedly wrote the "Csound Ezine". It appeared for several years and totals about 10 issues. A great and inspiring read to this day. Carrying the torch from Hans was James Hearon and Steven Yi with "The Csound Journal" which is up to about 17 issues now. They release four issues a year and the articles range from the practical to the esoteric, from a new look at the basics, to the exposition of the most

cutting edge topics. If there is something new that is possible with Csound, The Csound Journal is the place to find out about it.

There have been some important websites that give this international community a home. Our Csound page at SourceForge, managed by John ffitch is pretty important and continues to grow; and I have been maintaining *Csounds.com* for 20+ years now with the help of some fantastic Csounders and Berklee College of Music Students - Jacob Joaquin, Young Choi, Juno Kang, Greg Thompson, David Akbari, John Clements and Christopher Konopka. A few years back, a Csounder came forward, Cesare Marilungo, and helped to totally renovate and revitalize the site. His dedication and that of these Berklee alumni, have helped to make it a real "home-base" for Csounders everywhere.

We can't forget to mention the inspiration and insight that we have been able to draw from those great Csound instrument libraries and instrument collections. I had been collecting things since literally "day one", and many of them appeared first in "The Csound Catalog" (which has been updated this year by Christopher Konopka and Tito Latini and now contains over 14GB of categorized and organized working instruments in .csd format!) Some of my favorite "Classics" are the huge and diverse libraries created by: Josep Comanjuncosas, Steven Cook, and Hans Mikelson; and the Csound version of all the instruments from "The Computer Music Book" by Charles Dodge and Thomas Jerse; and, of course, the gem of the collection - Jean Claude Risset's "Instrument Catalog" from MusicV that was converted to Csound - this is truly an example of being able to stand on the shoulders of giants! Today's top Csound sound designers are: Joachim Heintz, whose examples in CsoundQt are incredible; and Iain McCurdy, who's "Real-time Instrument Collection" is so comprehensive and inspiring that it takes your breath away; Iain shows you how to do "everything"! And... it all sounds amazing! We owe these great sound designers so much, because it is their work that teaches each of us how "it" works.

And so... we have the tool (arguably the finest and most powerful synthesizer in the world). We have the instruments! We have the knowledge. We have the books, the journals, the papers, the research, the teachers and the curriculum. We have this amazing community. But... what we don't have... is enough music. If there is any *one* challenge that I would like to pass along to all of you here today, and to the entire Csound community, it is that we need to push ourselves to make more music with Csound. We need to use Csound in more of our compositions; (it doesn't have to *all* be Csound, but it *always* sounds better when you have added *some* Csound); we need to use Csound in more of our productions (add a

Csound reverb or ring modulation, or EQ); we need to use Csound in more of our game audio and sci-fi TV spots (I want to hear more background ambience, explosions, and alien voices created with Csound); we need to use more Csound in our installations (come on... Csound is the ultimate sonification tool); and we need to add Csound to our love songs (that's right, I want to hear Madonna, and Rhiana, and Prince, and Beiber, and Ringo, and Bono, and Janet, and Sting, and Usher using Csound on their next CDs). We all need to do more to turn our Csounds into Cmusic - to move from sound design to musical design.

The future of Csound is in the *music* that we make with Csound.

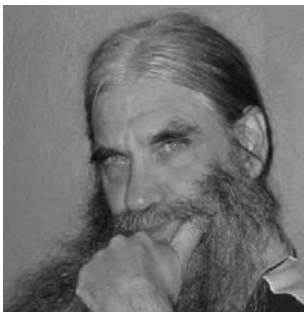
For me, Csound has always been the place that I could turn to for answers. The answers were in the instruments, in the code, in the emails, in the generous gifts of time and knowledge that this community has shared with me and with each other. In the beginning, I was fortunate enough to hang out with Barry Vercoe at the MIT Media Lab and when I had a question, I would walk into his office and ask him. I can't possibly tell you everything that I learned from him; but I have tried to pass along much of what he taught me through "The Csound Book", "The Audio Programming Book", “Csounds.com”, "The Csound Catalog", and most importantly, through my students.

In a way, Barry Vercoe trusted us with his "baby", and I think that we have raised a "child" that would make him proud.

Csound has brought us all together, and I am incredibly honored and proud to be associated with each and every one of you, and truly grateful for all that you have taught me. Thank you, for giving me this great opportunity to thank you.

HOW TO BECOME A CSOUND ROOT?

INTERVIEW WITH JOHN FFITCH



John ffitch has been at the forefront of Csound development for about twenty years. Despite long hair and beard, was never a hippie, he says in his biography. In this interview he talks about his way to Csound, and his view on the different periods of its development, including the ways for the developers to collaborate, before and after the Sourceforge platform brought a new standard.

How did you come in contact with Csound?

First a little background. I started writing (bad) music way back, but got very serious about it in the early 1960s, when I wrote a piano concerto late at night, every night after I was supposed to be in bed, and other works, like pieces for girls at school. However not being a pianist and only a self-taught recorder player, I could only hear my music in imagination. About 1970 as a research computer geek I thought about getting a computer to play some of it for me, but cosmology and astronomy took my time outside computer algebra and LISP.

When I got my chair in Bath and having done a lot of administration I returned to thinking about music again. So I lurked on net-news systems reading music material. As a result I took the sources of Csound from MIT. I had a Unix computer as my home machine so building it was reasonably easy. Never managed to work out how to use it but it was there. Must have been late 1980s or 1990. All this introduced me to a world I did not know, and led me to a conference in Greece celebrating Xenakis, and then to ICMCs.

What was your first participation in the development of the Csound code?

My general interest in music led me to lurk on newsgroups; really cannot remember which one it was but possibly comp.music. After some time with little understanding someone asked if Csound ran on a PC. As I said I had downloaded the Csound sources from MIT earlier, but it was a wet weekend and my software company (the owner of my unix computers) was developing and selling algebra software on a range of systems, including PCs, and so I had a C compiler for it, and time, so I tried to build Csound. Actually was not very hard. I announced this to the news group and that was all. A short time later the same question was asked so I replied along the lines of "yeah did that a few weeks ago". I then started getting emails asking me for it. I think the majority of the messages came from Dennis Miller at Northeastern University. I started looking into realtime output from a soundblaster and fixing compilation problems. All I did was on PCs/DOS at this time, but I did expand to Acorn and Atari over time. Perhaps I should add that my main job at this time was researching and teaching software, especially LISP and event simulation, with a small sideline in music software, like Rosegarden.

How did you become the main responsible developer?

By mistake? Out of the blue I got a phone call at home from Barry Vercoe (who I only knew by name) introducing me to Mike Haidar of Analog Devices. They had heard of what I had been doing on PCs and invited me at short notice to Boston and Norwood, just before Christmas 1994. This was the Extended Csound project, using a DSP processor to get real-time synthesis. I had a great time, and as well as Barry I met Richard Boulanger (DrB, Rick,...). I flew home on Dec 23, arriving (late) at Gatwick on Christmas Eve just in time for the last train home.

From this I visited Analog Devices Inc on a number of occasions, and increasingly I was working at weekends (and sometimes all night) with Rick, and making changes as he suggested. Over time I moved the code to ANSI C, added some opcodes, etc. I never consciously ``become the main responsible developer" but it sort of emerged that Barry was happy to let me deal with users, and he was working on Extended Csound; I suppose Rick pushed me into it. I invented things like version numbers, changelogs etc., and I was now working on Unix, Mac, PC/Windows, Acorn and Atari. At this time *de facto* I had complete control over sources, and what went in.

Who else has been on board at this time? How did development work without platforms like Sourceforge?

People sent revised code or patches, or complete opcodes and I checked them, ANSIfied them and incorporated in the next release. Releases happened when I wanted, or a major bug was fixed or on demand. There were a number of other contributors whose names can be found in the current sources and manuals, but I did the incorporation. Many people proposed many changes, and I tried to act as liberally as I could.

I have checked the files and there were 49 people credited in the sources before we went open source as contributing (list naturally available). Of those about eight stand out in my memory as contributing to the core (Richard Dobson, me, Michael Gogins, Matt Ingalls, Victor Lazzarini a little later, Gabriel Maldonado, Istvan Varga and of course Barry Vercoe) but this list is not complete. For example a student of DrB asked me for a feature in the score -- he contributed but his name is buried in a comment I think. DrB himself did not write a line of code but was active in pushing for improvements and developments. And there were others like Maurizio Umberto Puxeddu who was very active for a while, did a great deal and then vanished from our world. The breadth of actions can be seen in the mailing list, which was run from University of Exeter by a local user for many years. Too many people to catalog!

When did the Csound development move to Sourceforge and what were the resulting changes for you and the other developers?

We registered for SF on 2003 May 25 as Csound 5 was taking true shape. Apart of the licence change we had to have an accessible base and Sourceforge was the obvious solution at that time. Naturally the changes in work process was major. Personally I had never used any source control system, and not having the master sources directly on my computer was initially hard. But I quickly found it useful even between working on my desk machine, laptop or university desk machine. One had, on the other hand, to take care about submitting non-working code. But a good discipline.

I have over the years worked on many software projects from about 1970, but usually with one or two friends with whom I shared an office, or later a close relationship. Not seeing the co-workers was a change. But one learns. At least in Sourceforge I controlled who had what access!

As you mentioned before, you have a liberal way of retaining control. Do you believe in Csound as a pluralistic or even anarchic project?

I had not thought of labelling it, but I suspect anarchy is close to the truth. The main developers were/are all composers and that tends to influence what gets developed. Sometimes I get into less popular areas tweaking code and adding system-like things, but we are a mutual community driven by users' requirements. Personally I like writing code, and implementing other people's ideas is an entertainment for me; and I enjoy the dynamics of working with musicians.

Interview: Joachim Heintz

LICENSING CSOUND TO LGPL

INTERVIEW WITH RICHARD BOULANGER

Richard Boulanger, editor of the famous *Csound Book* (MIT Press, 2000) and host of the main Csound community site *www.csounds.com*, has not just accompanied every step of Csound since its „birth" in 1986. In this interview, he describes both, the years before this first release of Csound, and the complicated process of moving Csound to the GNU Lesser General Public Licence.

Richard, you actually worked with Csound before its first main release in 1986. Your piece "Trapped in Convert" was written in 1979. Please describe working with Csound at this time?

The spring and summer of 1979 were life-changing for me. So many things were coming together in my studies, my composing, and my career. I was working on my Masters Degree at Virginia Commonwealth University where I was the teaching assistant in Electronic Music and had 24-hour access to several labs filled with some incredible analog modular synthesizers. In the fall, I had a symphony for large acoustic orchestra and 2 Arp 2600 synthesizers commissioned and premiered by two different orchestras (I was one of the synthesizer soloists).

That spring, I was invited by Dexter Morrill, to spend a few weeks at the computer music lab at Colgate University. (At the time, this was one of the few places in the US with high-end digital-to-analog converters; and with a mainframe that had any sort of MusicN language installed on it. It was a DEC PDP10.) Not only did I have weeks of unlimited access to this incredible machine; but I had the great fortune to be working there together with another visiting artist—a recent Stanford/CCRMA graduate Bruce Pennycook. Bruce spent hours teaching me Stanford's Music10 program. He helped me to create the most amazing sounds I had ever

heard, and he laid the foundation for my deep understanding of how computers actually make music. With him, I was able to make the connection, translation, and transition from my experience and work with analog modular synthesizers to their digital counterparts. And later that same summer, I was invited to return to Colgate, and to work there as the "Studio Assistant" for Dexter's Computer Music Workshop. There I met, helped, and learned from a number of legendary electronic music composers; guys I had been reading about, and whose records I had been buying and devouring for years.

Dexter Morrill was not the only person offering computer music workshops in the summer of 1979. Barry Vercoe was also running a highly publicized and coveted set of workshops at The MIT Experimental Music Studio. They attracted engineers, programmers, and composers from around the world who would come for a month to learn about these new possibilities and work on systems developed at MIT to digitally transform sound and make computer music. There were about 20 of us in the class, but only 10 were composers and only these 10 would stay for the full four weeks. The first two weeks were focused on theory, hardware, and software (this was mainly for the engineers, but the composers were there too). Every morning, from 9am until noon, Barry Vercoe, (mostly from memory), would fill the walls (literally "fill" the walls - they were all "white-boards") of a very large MIT classroom, (we all sat side by side at long white tables in the center of the room), with equations, block diagrams, circuit diagrams, instruments, opcodes, algorithms, and code. We were literally surrounded with his work. And when he ran out of space, the walls would slide from side to side to reveal additional white-boards on which to write. It was breath-taking how much information, knowledge, and experience was pouring out of him. And I was doing my best to absorb as much as I possibly could! Each afternoon, we would have a distinguished guest like Marvin Minsky (the father of AI), or Max Mathews (the father of computer music), or Mario Davidovsky (pulitzer prize winning electroacoustic composer) to name but a few, present their research and creative work. During the second two weeks, most of the engineers went back to work, and the composers stuck around and applied their new found knowledge and understanding to the creation of new pieces that were premiered on the final day in a major, sold out, concert (yes, this was really "new" back then, and quite rare, and so there was a huge public audience interested in what we were doing at MIT, what the future of music would be). The concert was held in Boston's famous and beautiful sounding Kresge Auditorium. It was reviewed in all the Boston papers as well.

Along with some promising (and now quite famous) young composers, and some already established celebrities, I was accepted into this 1979 summer class. It was very inspiring to work with, live with, and hang out with this group. Strong and lasting friendships developed. We all lived in the MIT dorms, and we worked around the clock - five of us on the computers during the night shift, and five of us on the computer during the day shift. My twelve hour time slot was from 6pm to 6am. What was so great about the late-night shift was that most of the morning shift composers didn't usually show until about 9am or later, and so, I would always get at least 3 or 4 more hours of pretty much solo computer time each day. In fact, most in my group would drop off to sleep between 1am and 6am and the load on the CPU would lighten then too.

We were all sharing time and CPU cycles on the Digital Equipment Corporation (DEC) PDP11 Minicomputer (about the size of a refrigerator) on which Barry Vercoe had written music11, a newly revised, optimized, and significantly expanded version of his former, MUSIC 360 language (a language based on Max Mathews' MUSIC V) in RT11 Assembler. It was fast for those times (in fact it could do real-time computation of simple oscillators and envelopes), but with five or more of us rendering audio at the same time, (most of us would leave our jobs running for days because that's what it took to make even simple sounds back then), and even running our jobs at a 16k sampling rate, to lighten the load and speed up the turnaround, one waited a long, long, long, long time to hear a few seconds of often very bad, very loud, very weird, very nasty, totally unruly, and absolutely unexpected "garbage". A *long long* time. For most... too long. But for me... well…

I loved it.

This was "composer's time". Time to study. Time to read the manuals. Time to listen again to the Vercoe lectures (I had recorded them all on my portable "cassette tape" recorder). This was time to dream. And time to talk about new music with each other, and about each other's music. Time to learn from each other. Time to explore all those amazing new opcodes. Barry Vercoe's music11 was an infinitely large synthesizer, and I went through the manual trying to make sense, and sound, of every module (opcode) he offered.

I would use this time to design instruments (on paper), type them in from the DEC terminal (click, click, click, click - green phosphor), and imagine how they would sound when rendered. There was no mouse. I forget what editor I used - probably VI.

Barry called his software synthesis language "music11", naming it after the PDP11 computer on which it ran. His personal interest in the

PDP11 was to do cutting-edge research on computer-based real-time synthesis and signal processing. He demonstrated these systems for us. His visionary "RTSKED" program not only rendered in real-time, but supported a Common Practice Music Notation display, and a visual instrument design language, that looked and worked very much like Max/MSP or PD, with which one could design "instruments" graphically by "patching" together icons representing music11 opcodes on the screen. Yes, the future. And more, there was an electronic piano keyboard attached, so that one could actually play the computer!

All of this was at least 10 years ahead of its time, and was so incredibly exciting to all of us; but the quality of the real-time synthesis and the limited number of algorithms and opcodes that one could run, in real-time, served to shift my focus to music11 proper because I wanted to make *huge* and *complex* sounds. For years, I had been working with my Arp 2600 analog synthesizer, my Arp Sequencer, my Oberheim Expander, some PAIA and Aries modules, and a Sony 4-track tape recorder and TASCAM mixer (a pretty nice set up really), and I *loved* designing sounds, creating evolving textures, and delicate ambiences; but there wasn't a day that went by that I didn't wish I had a few more oscillators, a few more filters, some additional envelope generators, more voltage processors, modulators, attenuators, patch cords and mults. With music11, my prayers had been answered. I finally had all the filters, oscillators, and envelope generators that I could imagine - and then some! And not only did I have more "modules", but with music11, I could automate all of their parameters using Barry's line, expon, linseg, and expseg opcodes.

On the Arp I would use envelope generators (2) and oscillators (3) to apply dynamic qualities to my patches; but here at MIT and with music11, I was thinking in terms of time and space - auditory space, spectral space, physical space - clock time, musical time, and psychological time. Instead of patches that I would trigger and play from a keyboard, I began to think of sound objects, spectral mutation, complex contrapuntal textures, radiant new timbres, and the most acrobatic melodic and rhythmic gestures - all moving, morphing, evolving, breathing, and living. All of my instruments in "Trapped" were conceived as unique expansions of what I might have done if I had an unlimited Arp2600 synthesizer at my disposal. Well, in fact... I did.

"Trapped in Convert" was the musical result of that summer in paradise. OK, well it wasn't always paradise, in fact, that's how the piece got it's curious name. At MIT, they were incredibly proud of their floating-point Digital to Analog converters custom built by Analog Devices. And as I had stated earlier, high-quality DACs were pretty rare in

1979. Given that we were sharing this computer, and we were not all too clear about what exactly we were asking it to sing or play; there were many "crashes". When the computer crashed, there was always a printout on the huge (and noisy) line-printer/mechanical typewriter in the machine room. This printout read: "trapped 0 in convert...".

I think that filters were blowing up because we were "dividing by 0 all over the place". As you might imagine, by the end of the summer, as I was desperately waiting (always hours and sometimes days) for the final sections and sounds to render, praying that it would all run without a crash, and praying that what came out would be good enough for the concert, I began to feel like "I" was "trapped in convert". I remember walking over to the printer one evening, reading the printout, and saying to everyone - there's the title for my piece - "Trapped in Convert!"

As the days and nights passed, it seemed an even more fitting title, and I made adjustments to sounds and structure of the piece to make it an even better fit. For instance, the finals section of the piece was particularly composed to have a sense of being "chased" and, after a short silence (possibly caught?), and then breaking free. In the middle of the piece, I added a section where there were "samples out of range" that distort and sound like the speakers were tearing, all to enhance the sense of "struggle" with the machine. Audiences seem to get it, and have been getting into it for many years now. It is quite gratifying and humbling - that such a simple piece could have such a long life and inspire so many generations.

When I came back to MIT in 1986, after completing a PhD in Computer Music at the University of California in San Diego doing "Cmusic" with F. Richard Moore, Barry showed me a new project that he was working on - rewriting music11 in C. He invited me to join him at the newly opened Media Lab, to help with the testing and especially to use "Trapped" as a measure. I would meet him at the Media Lab at 6am every day (which meant leaving my house at 4:30am.) We would start downstairs on the 4th floor with a coffee (he taught me how to double-up the filters to make it even "stronger"), and then we would get to work. He would add new opcodes, give me a new version of the program, and I would design instruments to test each of the opcodes. It was so exciting.

We did this for several months, until all the opcodes were there and all sounded the same. We worked until "Trapped" worked and sounded identical.

To the user, Csound looked, rendered, and sounded exactly the same as music11. In fact, many of the manual pages for the Csound manual were taken from the music11 manual. In the fall of 1986, all of the music11 opcodes were rewritten in C and became the core set of Barry's new

computer music language - Csound. I was happy to be there at the dawn of a new era and to have been able to suggest some additional opcodes, like "loscil" that he added at my request. I am still suggesting new opcodes to this day.

I was happy to have been able to continue working on Trapped at the dawn of Csound. I fixed some things here, balanced some things there, tweaked the ending a bit. In fact, I will admit it now, for the 1979 premiere of Trapped, the ending was "spliced", but in the Csound version it is not. Back then, the entire concert was played from analog tape. (Quite ironic don't you think?) There had been a few too many "trapped 0's in convert" the days leading up to the concert, and we could not get a complete render of the piece done without one of my "colleagues" crashing the PDP. (It actually took days to compute the four minutes back then.) We had a clean render of most of the piece from a week earlier already on tape, but we did not have the *whole* piece. Time was short. We decided to just compute the ending (which took a night), and then bounce that to tape and splice it, just minutes before the start of the concert. In fact, the silence prior to the last set of sounds was something we added from the "cutting room floor" to give it a little "breath".

Now you know the story of Trapped in Convert, how it got it's name and how it came to be. The "Trapped" that everyone knows today. The one that most Csounders use to benchmark their CPU, or test the latest build of Csound, was indeed the very first "Csound" piece, but for sure... it will not be the last!

Csound can be considered to be "Open Source" at that time. But was it legal for anyone to distribute and use copies of Csound?

The story of The Csound License is quite convoluted. As I understood it, there was initial funding from an NSF grant that helped to support Barry Vercoe's original work on Csound. And this grant stated that the results of this work should be open for "educational and research use". That's nice, but this "distinction" left quite a number of us, those who actually "used" Csound, quite confused. What made it worse was the fact that the copyright also stated that one needed "permission" from MIT to do anything else with Csound. Not particularly "open" right? Many of us wondered if we were allowed to "compose" with Csound or "perform" with Csound, or "process recordings" with Csound, and we were very concerned when we thought to make and release records and CDs using Csound. These all seemed to be "commercial" uses of the program that might not fall under the "education and research" label.