TEACHING SOFTWARE SYNTHESIS THROUGH CSOUND'S NEW MODELLING OPCODES

Richard Boulanger
Berklee College of Music, 1140 Boylston Street, Boston MA 02215-3693, USA
(rboulanger@berklee.edu)

John ffitch
Media Technology Research Centre, University of Bath, Bath BA2 7AY, UK
(jpff@maths.bath.ac.uk)

Abstract:
The computational power of today's computers brings the most powerful signal processing and synthesis tools into the home studio. For less than $2000 you can have the equivalent of the IRCAM ISPW or a NeXT running “SynthBuilder” on your battery powered PPC or PC Laptop, and all the “essential” software tools and utilities can be had commercially at low cost or are freely available over the Internet. Thus we are seeing that independent composers and freelance artists are seeking out this software, and placing demands on the designers for ease of use and general utility. This paper concentrates on one of the remaining problems which limits the widespread use of the technology, and that is the complexity in specifying every feature of a composition, and in particular describing the detailed timbral evolution of each note. We explore the new facilities in Csound which address this problem.

1. Introduction
Cost is no longer the issue. For very little money you can have a machine as powerful as the previous generation of research equipment, but on your battery powered PPC or PC Laptop, and what is more, software tools and utilities are freely available over the Internet or at worst cost only small sums commercially. Thus we are seeing that independent composers and freelance artists are seeking out this software, and that is placing demands on the designers, as these new users demand ease of use and request high utility.

One of the remaining problems which limits the widespread use of the technology is the complexity of specifying every feature of a composition, and in particular describing the detailed timbral evolution of each note. With the forthcoming publication of a detailed pedagogical book on Csound [Boulanger 1998], the authors are concerned that the initial enthusiasm may die before the skills and excitement in crafting detailed musical sounds and moments are fully developed.

While not disparaging the necessity for accomplished practitioners to learn these skills eventually, we describe here the attempt we have made to provide a comprehensive set of synthesis models, which can satisfy the immediate requirement for “cool presets”, and so get the beginner started without too much investment in learning, and at the same time have sufficient control for exploration and variation. Thus the student of software synthesis can start to experiment with timbre from a secure base but with a wide number of interesting possibilities. The catalogue of “software synthesis presets” we have constructed include simple examples of “classic” synthesis techniques such additive, subtractive, FM, waveshaping, and granular, and these are already available. To these we now add physical models and physically inspired models.

This paper is in some ways a continuation of Boulanger [1992], as we have the same aim of improving the teaching of synthesis to the rising generation.

2. Hardware Synthesisers and Software Synthesis
For many years hardware synthesisers have delivered users with pre-packaged versions of the synthesis techniques that have been explored slightly earlier in software synthesis. At certain times some methods have become fashionable, and become ubiquitous. For example FM synthesis [Chowning 1973] became the basis of much popular music through Yamaha synthesisers.

The latest synthesis technique to reach “fad status” at both the ICMC and the Tower Records level is the simplification of the physical generation of sound typified by the digital waveguide [Smith 1987]. This technology
has been embraced by at least the Korg Prophecy, ZI, WaveDrum, the Yamaha VL1, VL7 and VL 70m, and the Windows95 Native Signal Processing Software Synthesiser Reality from Seer Systems. What these commercial hardware and software synthesisers offer is ease of use, and clearly packaged examples of the methodology. This is in contrast to the arguably more powerful general purpose software synthesis systems that remain difficult to use, despite, or possibly because of, their flexibility. In the commercial form, they still primarily offer the user hundreds of presets and that most of the “tweaking” of these unique and advanced synthesis models is in the areas of modifying resonant low-pass filters on the output of the algorithms, modifying overall amplitude envelopes (again on the output and not on the parameters of the model) and changing effects (reverb, chorus, delay), on the output of the model. In effect most of the editing here is to effects, leaving the model, which the user still does not understand, untouched.

3. Waveguides in Csound
The waveguide synthesis method is available from within Csound in a variety of fashions. These range from the “preset” codes to detailed instruments. We will consider three levels of physically inspired models of sound creation and consider how they are available, and more importantly how they can be used as part of a graduated learning system which will eventually open up the rich possibilities of software synthesis.

(a) Physical Model Presets
There is a set of physically based synthesis algorithms due to Perry Cook [Cook 1996, 1997], which have been re-coded in the Csound format. These provide within a single opcode approximations to the sounds of a clarinet, flute or brass instrument for example. The opportunities for personalisation are limited, but not totally lacking. The documentation suggests ranges for the various parameters, such as reed stiffness in the clarinet model, but the adventurous student could explore the effects of these parameters, as well as going outside the recommendations to see what happens. As an example of the style of these opcodes consider the waveguide clarinet, wgclar.

instr    1
    ;;          amp,      freq, stiff,att,dtk,nGain, vFrq, vAmt, vTab
    a1      wgclar  p4, cpspch(p5), -0.3, p6, 0.1, 0.2, 5.735, 0.1, 1
out     a1
endin

This uses the recommended default values, and as such is the simplest such use. The sounds generated are lacking in any character as a collection, but it is easy for a student to consider modifying the vibrato characteristics, the reed stiffness and so on.

Some of these preset-style opcodes admit much many complex user controls. For example the opcode wgflute, in addition to amplitude and frequency, has controls for the time to build the jet, and the time over which the jet declines, controls for the amount of added noise, frequency, amplitude and wave form of the vibrato, and control of the reflections at either end of the tube. A student bored with the simple instrument has a wide variety of parameters to adjust which have the added advantage that they are related in an intuitive way to the physical world. Not quiet so physical, but in a similar way the fmbell opcode (and other similar opcodes) can emulate the FM instrument, but being based on four external wave forms, as well as a vibrato shape, an adventurous student could explore by using alternative waves with very little Csound experience.

(b) Constructing Waveguide Instruments from Components
The preset physical models are useful simple sounds, but one of their main uses is to encourage the student to look more closely at the underlying modelling concepts, and eventually to be driven to create their own instruments based on these models. The basic clarinet model is well know as a combination of delay lines and filters, and so the student can take the flow diagram and code it using the standard Csound opcodes. There are a number of choices which have to be made, such as the type of filters used, but it is a reasonable exercise for the student to take the basic figure
and produce [Mikelson 1998]

```
instr 2 ; Clarinet Instrument based on Cook's Clarinet
  areedbell init 0
  ifqc  =  cpspch(p5)
  ifco  =  p7
  ibore =  1/ifqc-15/sr
  kenv1  linseg 0, 0.005, 0.55+0.3*p6, p3-0.015, 0.55+0.3*p6, 0.01, 0
  kenvibr linseg 0, 0.1, 0, 0.9, 1, p3-1, 1 ; Vibrato envelope
  kemboff = p8 ; Adjust reed stiffness.
  avibr  oscil 0.1*kenvibr, 5, 3 ; Breath pressure.
  apressm = kenv1+avibr ; Lowpass Reflection filter at bell
  arefilt tone areedbell, ifco
  abellreed delay arefilt, ibore ; The delay from bell to reed.
  ; Back pressure and reed table look up.
  asum2  = -apressm-0.95*arefilt-kemboff
  areedtab tablei asum2/4+0.34, p9, 1, 0.5
  amult1 = asum2*areedtab
  ; Forward Pressure
  asum1  = apressm+amult1
  areedbell delay asum1, ibore
  aofilt atone areedbell, ifco
  out   aofilt*p4
endin
```
c) Hybrid and non-physical sounds
With the underlying ideas of physical modelling it is possible to build alternative sound generators. As a simple example of this we present the `repluck` opcode of Csound which takes the basic structure of a plucked string model, and allows the “string” to be excited by any audio signal in addition to a simple pluck, as discussed in Boulanger [1985]. This is inspired by the Aeolian harp, a sound which is under-represented in computer music. This instrument is an encouragement to experimentation outside the range of traditional instruments.

```
instr 3
  a0 in
  a1  repluck p4, cpspch(p5), p6, p7, p8, a0
  out  a1
endin
```

When excited with speech in particular this acts as an interesting resonator.

We would encourage students to look at other possible filters, such as the `nlfilt` opcode [Dobson & ffitch, 1996], or any other extreme ideas from the standard collection of opcodes.

### 4. Future Directions
What we have presented here is only a beginning. In order to maintain the seamless transition from preset sounds to full scale physical model inspired synthesis there is a need to develop Csound in new ways. What is needed is a set of new primitive opcodes which take their inspiration from SynthBuilder [Staccato] for example, which would allow the creation of instruments from delay lines and filters in a natural way. It is too early to report on progress made in this work, but we are starting the process of deconstructing the preset physical models which are largely taken from
the designs of Cook. It is easy to identify a delay line with flexible treatment of the fractional delay filter as a basic component, as well as a variety of filters. We still need to construct the total model.

5. Conclusion
When designing a set of models particularly focused on pedagogy, one is faced with the question of how much control to allow. Having favoured the belief that more control is better, we have still managed to design opcodes with recommended ranges of use and while maintaining the Csound programming model resort to tables and function generators where they are appropriate. Thus, we believe that we have constructed a set of sounds, only a few of which are given here, which are useful for educational purposes, and will allow a student to compose and investigate the electronic medium and yet require but a minimum of technical or mathematical background. The goal being that these sounds and models can form the core of the student's first steps to finding their individual “soft-voice”. What is more these models lead the student to explore new combinations and expand them with the inclusion of other opcodes and controls.

References


Cook, P. R., *Physically Informed Sonic Modelling (PhISM): Percussive Synthesis*, ICMC 1996 p228-231


