One and One and One is Three: Unification of Control, Synthesis, and Display

Steven Backer
Center for Computer Research in
Music and Acoustics (CCRMA)
Dept. of Music, Stanford University
Stanford, CA 94305
+1 650.917.9038
sbacker@ccrma.stanford.edu

1. INTRODUCTION

This paper discusses the design and implementation of a musical "instrument" project for the MUSIC 250A course, from various perspectives. It is intended to be a project report and assumes the reader is familiar with many aspects of the project.

2. MOTIVATION / BACKGROUND

"Traditional" computer music instruments are more or less the sum of a number of other discrete tools – computers, displays, control interfaces, speakers, and the like. Despite functioning as one, these elements usually remain physically disjunct. In certain situations, not having the sound come directly from what is seen as the instrument can significantly affect the audience's perception of a performance. Furthermore, the tangible and physical aspects of computer music instruments often involve many long wires, laptop computers/displays, patch bays, pedals, and other cues to an uninformed audience that something *besides* music might be happening on stage.

That is not to say that one cannot redefine his or her definition of a musical instrument – some of the most amazing computer music instruments have filled entire rooms! However, there is still a tendency to make these new instruments more like the ones we are used to – the violins, guitars, and pianos that children see real musicians playing. It is this need to reconcile computer music instruments with traditional instruments that this project attempts to address.

This project consists of various technological puzzle pieces, assembled in a manner that hopefully can create sound that is both expressive and flexible. It was not intended to be easy to play, just *easier*. It also seems that more often than not, the wider an expressive range an instrument has, the more difficult it is to play, and much practice is needed to learn how to play it well. So, it is expected that the outcome of this project be a little unclear at this point.

Most of the pieces of the instrument are clearly depicted in the figure at the end of this report. The reader should reference this figure along with each subsequent section from here on. One should quickly notice that at the core of the instrument is an embedded DSP. This is a very important implementation decision. The most notable reason is that DSP chips "are far more efficient in handling low-latency input/output"[2] (This is really outside of the scope of the paper.)

An implementation using instead the Atmel AVR for synthesis, followed by a DAC, was considered as an alternative to the embedded DSP solution. In short, the ATMega16 is not designed for real-time signal processing of this sort. Additionally, quality DACs seemed to be a bit pricey for this project. The AVR did prove to be the perfect solution to the analog control interface, though. Not only is it a familiar platform (which the whole class uses), but also it is a useful debugging tool.

3. CONTROL

3.1 Mapping

The granular synthesis technique (GS) is an interesting one to implement in an instrument, partially because the mapping of concepts into the algorithm has already been thoroughly defined by people such Xenakis, Roads, and Truax. It is the *access* to these parameters that plays a large part in what the the granular resynthesis will sound like. Controlling GS algorithmically with a script, versus in real-time with physical sensors, significantly influences the performers interaction with the sounds; both methods have their pros and cons.

In this project, the parameters of a GS algorithm are made available to the performer in the physical world, as buttons, knobs and sliders – simple, yet "classic and effective" control sensors. The first six parameters in the table below are existing arguments from STK function calls. The seventh - "Grain Lock" - is a device added above the STK generator in the code hierarchy to allow for control of sound I/O in relation to the algorithm. The granulator continuously updates a grain buffer from the real-time audio input, *unless* the blue button is pressed, in which case the algorithm locks onto a fixed grain source.

Table 1. Mappings from Physical to Virtual

Parameter	Physical Type	Virtual Values
Grain Duration	Knob	0 – 255 ms
ASR Envelope	Knob	0 – 100
(Grain Pointer) Offset	Knob	-128 to +127
Delay	Knob	0 – 255 ms
Stretch Factor	Knob	1X - 10X
Randomness	Slider	0.0 – 1.0
Grain Lock	Button	-

3.2 Bridge from AVR to DSP (UART)

One of the major technical challenges in this project was to determine a method by which to get the digitized sensor data from the AVR processor into the DSP and accessible inside of its program. This was successfully accomplished via a UART link between the AVR and the DSP EVM. Not only did this require an understanding of the external UART IC's on both sides of the link, but also programming of specific software routines for each platform. From the AVR, the UART library was found very easy to use, and a few simple function calls got the sensor data on its way. However, the coding solution on the DSP side proved more evasive. Many registers on off-chip UART interface had to be programmed (by the core DSP program), and the process scheduling had to be well understood.

Even after getting data to correctly transfer from chip to chip, the DSP code still needed a method to determine which data value applied to which C++ variable in the granular synthesis code – that is, a protocol. A simple, yet effective solution turned out the work just fine. Both UARTs were run at a fixed rate, and the seven sensor values were continuously resent one after another. In between each group of seven, an arbitrary two byte header was inserted. In the DSP code, all 9 values were read into temporary variables, and then easily sorted and mapped based on the location of the header in the stream. Of course, this is not the most efficient implementation, but function just fine for these purposes (and still with low latency).

The baud rate was originally set to 9600. This is probably a bit slow, and may be a bottleneck in the system. Higher rates need should probably be used, when time permits. Other options considered for the interface were SPI, and a direct memory interface.

4. SYNTHESIS

4.1 DSP Synthesis Engine

The synthesis of the audio samples occurs in the DSP synthesis engine, driven by a loose framework of code compiled specifically for this project. The behind the scenes situation is pretty ugly, and the details won't be discussed here. Just as an idea, though, here are some of the issues and challenges dealt with over the course of the project only within the DSP synthesis engine: Coordinating an interrupt driven real-time audio handler, the use of fixed-point numerics, meeting real-time timing constraints, creating periodic interrupts, assigning optimal interrupt priorities, demystifying the UART interface, and memory allocation. As is true with any computer, it was not a simple task to program the DSP (and have it actually work)!

4.2 Real-Time Granular Synthesis via STK

The real-time synthesis algorithm used within the DSP code is based on the STK "Granulate" Generator [3] class. Like the STK, this asynchronous granular synthesis algorithm itself is one that has been used extensively at CCRMA (and elsewhere) before. The STK version is one implemented by Gary Scavone, based on the MacPOD program [1] by Damian Keller and Chris Rolfe, which is a derivative of Barry Truax's original 1988 POD system.

The DSP code, at its core, consists of a port of the STK generator class to code specifically compiled for the C5410 EVM board/chipset. Many of the variable types, and much of the structure of the C++ code had to be altered to accommodate various idiosyncratic aspects of the compiler[1] used to generate the DSP program code. For example, not having use of vectorized audio and STK frames objects required some non-trivial recoding of overall program structure.

An extremely important (and challenging) implementation detail involved the task of altering and adding to the STK code such that it could run in real-time, capturing audio from the environment rather than reading from a file. Furthermore, many of the control parameters needed a method of updating *while* the algorithm was in the middle of execution. A top level of abstraction exists in the code to handle the real-time audio and control I/O. This is perhaps one of the most enabling (and limiting) aspects of the code – it could probably still use some work. Its essentially where the software diverges between an embedded instrument and one running on a general purpose computer.

Certainly one could use any synthesis model, or for example, a different instrument from the STK. One of the goals of this project was to lay that foundation, rather than getting too tied down to a specific algorithm. Nonetheless, focus, and a "guinea-pig" algorithm was needed – enter Granular Synthesis. Of course, hearing an inspiring work like Truax's *Riverrun* makes the creation of a real-time granulator very enchanting, too!

5. DISPLAY

Auditory display is the final element of this instrument. At the back-end of the DSP is an audio DAC, followed by an amplifier and speaker. The current implementation utilizes what was available; however, one could certainly upgrade or replace any of the above components with others of varying quality. In other words, it sounds ok with the parts used, but of course higher quality audio components can be used to improve sound quality.

Physically, the instrument housing itself is somewhat arbitrary – a plastic toy Cinderella carriage! Of course, one is tempted to choose a housing with good acoustic properties, appealing aesthetics ,etc., which inevitably leads back to all of the existing acoustic instruments that have been perfected over hundreds of years. This is ultimately a whole other can of worms to deal with in the context of a unified instrument, and this was not extensively addressed in the limited time given.

Nonetheless, there is some aesthetic appeal to the toy carriage. Its clear plastic shell reveals beneath it a complex tapestry of wires and circuit boards, in stark contrast to the innocent, playful exterior of a children's toy. Long used in music, the Cinderella metaphor remains relevant, as the conjured images of transformation (to a grander state) parallel the changes undergone by the small grains of sound as they become a more grandiose sonic event. Additionally, cheap plastic was easy to work with.

6. FUTURE WORK

Of course, since this project has only been underway for less than one academic quarter, there are a number of issues and improvements to be dealt with. In short, the major issues to resolve in any future work include: Getting rid of the power cords, extending algorithm implementation to achieve higher

grain densities, fixing miscellaneous bugs, figuring out how to flash the DSP board semi-permanently so it doesn't need to be reprogrammed (arbitrary, but time consuming), and exploring other platforms that may be more suitable (e.g. A floating point processor with multiple A/D channels).

7. CONCLUSION

In the end, the unification of the different aspects of a computer music instrument turned out to be a challenging yet intriguing topic to explore. While there is still much room for improvement in the instrument I built, the groundwork has been laid for a flexible, programmable, musical instrument that I hope to continue learning how to play.

8. REFERENCES

[1] Keller, D. & Rolfe, C. (1998). The Corner Effect. *Unpublished paper.* Burnaby, BC: Simon Fraser University

[2] Smith, J.O. (1991) Viewpoints on the history of digital Synthesis", in *Proceedings of the 1991 International Computer Music Conference, Montreal*. 1991, pp. 1-10, http://ccrma.stanford.edu/~jos/kna/

[3] http://ccrma.stanford.edu/software/stk/classGranulate.html

[4] Trueman, D. and P. R. Cook. "BoSSA: The Deconstructed Violin Reconstructed." in *Proceedings of the International Computer Music Conference, Beijing*, October, 1999.