The Gluiph: a Nucleus for Integrated Instruments

Sukandar Kartadinata
glui
Hagenauerstr. 6
10435 Berlin, Germany
sk@glui.de

ABSTRACT

In this paper I present the gluiph, a single-board computer that was conceived as a platform for integrated electronic musical instruments. It aims to provide new instruments as well as existing ones with a stronger identity by untethering them from the often lab-like stage setups built around general purpose computers. The key additions to its core are a flexible sensor subsystem and multi-channel audio I/O. In contrast to other stand-alone approaches it retains a higher degree of flexibility by supporting popular music programming languages, with Miller Puckette's pd [1] being the current focus.

Keywords

Musical instrument, integration, single-board computer (SBC), embedded system, stand-alone system, pd, DSP, sensor, latency, flexibility, coherency.

1. HISTORY

1.1 Sensor to MIDI - The SensorLab

While work on the gluiph in its current form started in 2001, the original idea goes back to the early 90s at STEIM, Amsterdam. At that time their main line of development were instruments based on the SensorLab [2] which basically is a single-board computer that converts sensor signals to MIDI. The interesting point was that its CPU was not just used to implement a mere data pump but to run an interpreter with an event-driven programming language called SPIDER, which was just powerful enough to render an additional PC on stage unnecessary, instead driving the MIDI gear directly. It was also small enough to be clipped on a belt, so the sensor lines could be kept short for better noise immunity.

1.2 Integrating Sound – The SoundLab

The early 90s were also the time when Motorola's 56k DSP chip was at its peak, being used in many commercial synthesizers, samplers, and effect processors as well as academic projects like CNMAT's Reson8 [3], CCRMA's Frankenstein [4], or the extended HMSL [5].

STEIM was introduced to the 56k in '93 by Steven Curtin [6] which lead to the idea to expand the SensorLab with such a DSP, so that one could design sound patches on a development PC and upload them to a stand-alone box in a similar way to the above SensorLab concept, thereby also getting rid of the MIDI rack. This project was dubbed the SoundLab, where, after Curtin left, I took responsibility for adapting the hardware while SPIDER was supposed to evolve to DSPider.

1.3 NSP – The PowerMac

However, with '94 came the PowerMac and its promise to do all sound processing on a general purpose computer. While initially not quite up to it one was right to assume that eventually it would, which for STEIM meant to abandon the SoundLab project. Since then native signal processing (NSP) has slowly become the main choice for most people to develop their instruments. Simpler Sensor to MIDI devices were released as now the PC could do the event as well as the sound processing.

1.4 Modern DSPs - hardMAX & mtronix

Still, for many years, embedded system held their advantages, and so certain advances in DSP development combined with on-going problems of PC-based systems lead to the hardMAX proposal on the Max/MSP mailing list in '99, with the basic idea of porting this software to an embedded device. However, this was largely rejected as G3 PowerBooks had become the center of stage setups, and hardware been declared as fetishism.

Due to the high development costs work on the SoundLab was stopped at this point, but only until in '01, during some projects for the Berlin company mtronix, a new strategy emerged that seemed feasible. There, the access to development tools and the shared system requirements allowed the design of a new musical instrument platform – the gluiph.

Before covering this in more detail though, I think it is necessary to explain the motivation in pursuing all this.

2. MOTIVATION

2.1 A DSP-PC Comparison

The justification for building embedded systems seems to have gotten more difficult. Ten years ago one could have come up with the following lists of advantages: latency/timing, processing power, size/weight, reliability. Today, things look different.

Latency and OS-related timing issues have largely been solved, provided one makes careful choices about audio interface and operating system, and maintains a certain amount of "hygiene" on the system. The MIDI bottleneck can still be a problem though, but newer sensor hardware has been released that communicates thru better interfaces (USB, OSC over Ethernet [7][8]), which should at least solve bandwidth problems. Only if zero-latency is required like in digital mixing environments, DSPs can hold their ground.

For processing power things have changed dramatically. At a time when PCs were still confined to non-realtime work, a 56k could easily shoulder reverb algorithms or many voices of sample playback. Now, CPUs with up to 3GHz make it possible to run complex instruments or complete studio setups on a standard PC. DSP-based systems have to use a (massive) parallel processing approach to defend their performance edge (e.g. Kyma [9], Scope [10]).

But then those devices are hardly smaller than a PC or even require an additional one for control purposes. At the same time laptops have replaced the desktops and their big monitors, so touring the clubs got even easier than with the then ubiquitous 19” rack. Only if even smaller systems are required, DSPs have an edge.

The reliability issue is debatable of course. After all we are dealing with complex systems in both cases, so there is always room for unexpected behaviour. And in fact, digital mixers, for one example, do crash at times. Still a point can be made that the additional complexity of a full-fledged operating system, that for the most part was not designed with musicians in mind, increases the likelihood of problems. At least I hear far less about catastrophe scenarios involving hardware equipment than with PCs. Concerning hardware reliability, embedded systems can be designed to provide better stability, e.g. sturdier connectors where you don’t break the motherboard when someone yanks on the cable.

Still, the bottom line seems to be that the advantages of DSP based systems have become somewhat marginal or confined to niche applications. So what is really left to justify the by no means small development effort invested into the gluiph project? The answer lies in the possibility of integration.

2.2 What defines an instrument?

I would like to start this discussion with a picture of a typical stage setup.

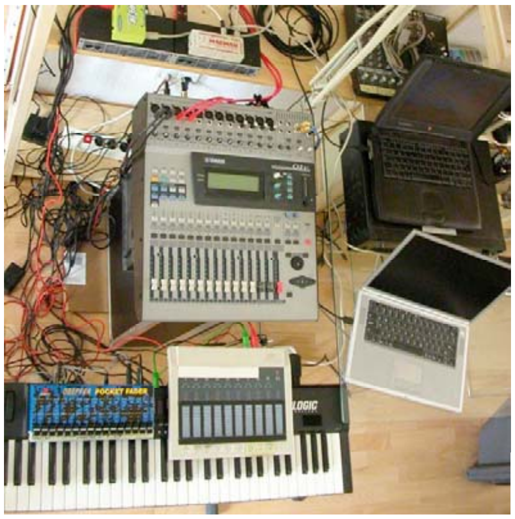


Figure 1. Two laptops, two audio interfaces, a master keyboard, two fader boxes, two MIDI interfaces, a mixer, 8 power supplies, 20+ cables

So is this what I want to call “my instrument”? Some might ask why not, but to me it looks more like an experiment from a laboratory. While there is sure nothing to say against being experimental in creating and performing one’s music, it is the goal of the gluiph project to derive at more mature instruments with a stronger identity. Integrating the defining components into a single physical entity is an important step in this direction, and not just a matter of aesthetics and easier installation (although those are by no means to be underestimated).

Flexibility, the key advantage of PCs, is a powerful concept for exploring the vast possibilities of today’s sound offerings. However, if a musician eventually wants to master his instrument, there comes a point when a decision has to be made about consolidating the existing setup, rather than getting lost in an ever-shifting environment. This reasoning follows the notion that reduction is an important phase in defining one’s means of expression. And that power has to be traded with nuance.

Of course, even a violin needs a bow, and an electric guitar its amplifier and loudspeaker. So not everything can and should be integrated. However, there are designs where the further integration of speakers can be advantageous (e.g. Nic Collins’ trombone [11]). In any case this exemplifies how the loudspeaker marks one of the key differences between electric and acoustic instruments, for which the “interface”, the “processing”, and the “speaker” are all intrinsicly embedded within their physical structures. To achieve such a degree of coherency for electronic instruments is one of the main goals of the gluiph project.

2.3 Related developments

Before describing the gluiph’s technology a few parallel development efforts are discussed that in one way or the other are related to its approach. Concerning the main concept of having a stand-alone music computer that receives its programming from a host PC, there are quite a number of commercial devices available:

Clavia’s Nord Micro Modular [12] also shares the gluiph’s size, however its sound generation is limited to virtual analog, i.e. mostly subtractive synthesis. More flexible are Creamware’s Noah [13] and Manifold’s Plugzilla [14] with their broader range of available DSP modules, where the latter relies on VST plugins. Both are 19” rack modules though, so they are hardly smaller than a laptop. Another device of that format is Sound-Art’s Chameleon [15], however it requires 3rd-party developers to write modules in DSP assembler. The same applies to the Death Synth by Noah T. Vawter [16] which again is smaller and was designed with PDA control in mind.

However, none of the above aims at integration but expects outside control thru MIDI, with all its limitations.

Others have suggested that, after the laptop, the PDA itself will be the next device to reach performance levels that will allow sound processing. However, currently the key problem is their missing floating point support which makes it difficult to run existing music applications. Wether this will change in the near future is rather unclear. The same holds true for the majority of commercially available Linux-SBCs.

3. TECHNOLOGY

3.1 Hardware

As mentioned in the introduction the idea to pursue the gluiph project was renewed while working for mtronix, a small Berlin company which mainly develops precision measurement instruments. They serve an industry where PC-based solutions are out of the question mainly because of reliability issues, operability requirements, and size constraints. Add to this the need for a powerful processor and realtime capability and one gets pretty close to the requirements of a musical instrument.

The key components of their boards were a Philips TriMedia CPU, SDRAM memory, Flash memory, programmable logic, PCI interface, and various sensor sections depending on the project. One interesting thing about the TriMedia is that it was designed as a multi-media processor with set-top boxes in mind, so this seems like a strange choice for measurement devices. For mtronix though it made sense as image analysis is one of their main applications, so they could make good use of the integrated video co-processing units. Video analysis, of course, is also of interest for controlling music software.

Another important reason for choosing the TriMedia was the fact that its core employs a very modern VLIW implementation with multiple execution units that are scheduled at compile time to keep the design light-weight yet powerful. As a consequence assembler programming is no longer feasible due to all the complex scheduling decisions to be made. Of course this approach requires a capable compiler, and in fact the one provided leaves nothing to ask for, while also complying to the full C/C++ ANSI standard and packing a well-tuned set of libraries.

Finally, and this with regards to the gluiph, the TriMedia features a 2in/10out-channel audio interface, so not much was missing to attempt a modification of the mtronix design towards a musical instrument platform. This was basically implemented by adding converters for the audio section and choosing a sensor subsystem. In addition the PCI connector was replaced with a USB port for a more appropriate host connection.

Also worth noting is the role of the programmable logic chip (CPLD). While many sensors provide an analog signal that can simply be connected to the sensor ADCs, others transmit digital data thru serial interfaces using a variety of protocols and baud rates. The CPLD can easily be adapted to those so no extra micro controllers are required. It can also be told to drive displays and relays or include precise timers for measuring ultrasound time lags. Towards the TriMedia it connects thru a high-speed interface that can be operated synchronously to the audio frame rate. MIDI, of course, is totally out of the game.

3.2 Software

The main premise in designing the software concept for an embedded music system was to do away with assembler programming. While in the past I had enjoyed tweaking DSPs for maximum performance, this was clearly no feasible approach any longer, mainly because of the lack of flexibility and the maintenance effort required.

The second decision was to avoid writing yet another patcher or even coming up with a whole new music programming language. Instead the plan was to port existing software, not only to reduce the development effort but also to benefit from the associated user bases. This, apart from the integrative concept, is the main difference between the gluiph and the other DSP based solutions mentioned above.

The third aspect emerged after the hardMAX proposal and its rejection, which was to rely on open-source software, thereby also following the general movement that was starting to grow at the time. The first candidate here was CSound as it could be operated from a command line in a non-realtime mode, which facilitated first tests for general operability. In fact, the port was completed literally in less than a day, and displayed performance levels comparable to a G3 PowerMac which ran at twice the CPU clock.

Rather than adapting CSound for realtime operation, it was then decided to port pd, partially because of CSound's more restrictive license, but mainly for personal taste and because Miller Puckette was encouraging the project. The main modifications to the source dealt with the MIDI implementation which was replaced with a direct link to the sensor subsystem. Other (minor) changes included glue code to the audio drivers as well as to the file system. Finally some performance optimizations were necessary, e.g. in handling the double precision arithmetic required by pd.

However, no special sensor-related external objects were written, rather the sensor signals are relayed to pd thru its standard send/receive objects.

3.3 IMPLEMENTATION EXAMPLES

3.3.1 *The SKRUB (for Richard Barrett)*

The SKRUB at first glance might not be the most spectacular sensor instrument, as it is "just a keyboard". However, it illustrates the effect of integration that can be achieved with the gluiph as it basically boasts the same functionality as the setup shown in Figure 1. There the PowerBooks run Max/MSP resp. LiSa patches that are triggered from a MIDI master keyboard and further controlled by two fader boxes as well as some foot pedals.

For the SKRUB the fader boxes were replaced with a row of trackpads, that have the advantage to be within easy reach from any position on the keyboard. They also add two dimensions of control – sideway movement and pressure (actually coverage).

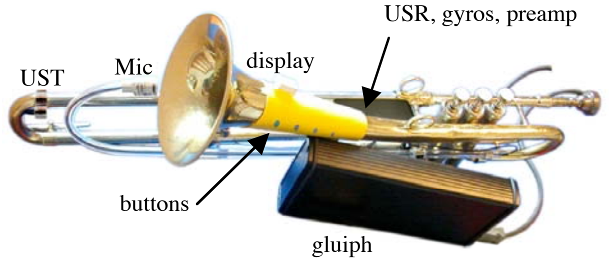
The keys themselves differ from ordinary ones by supplying continuous vertical position information for every key, extending the "trigger with velocity and aftertouch" scheme of standard keyboards. Thus a key can be used to e.g. scrub along a sample. Or a set of keys to knead thru a graphic equalizer.

For both sensor systems the CPLD proved very helpful. For the trackpads a 9-channel PS/2 interface was implemented, while the linear position sensing for the keyboard required a 61-channel frequency counter to measure the frequency deviation of the inductive sensor circuits. All key positions are sampled at 345Hz (fs/128) while trackpad coordinates are measured at 115Hz (fs/384, limited by PS/2 speeds).

The SKRUB also includes a 4-channel audio output for which the PowerBook setup requires external interfaces. Finally reverb was integrated, although not as software within the gluiph but thru an additional chip.

The control interface was kept at a minimum, providing only one LED per trackpad, its brightness resp. color reflecting the current levels of the parameters it controls. An additional button is used to switch to patch/bank select mode, however no extra number keys are used for selection, rather the keyboard keys take over this function.

3.3.2 *A meta trumpet for Rajesh Mehta*



This trumpet is an acoustic/electronic hybrid instrument. It is the first gluiph-based design and only employs a small number of sensors. The center box contains a 2-D gyro sensor that is used to measure yaw and pitch. The yellow leather sleeve around the horn houses a small display and a set of simple touch buttons that select different operating modes. The trumpet also stands out thru its trombone-like slide that allows a pitch change of a minor third. This extension is measured with an ultra-sound system.

An important aspect for Mehta was wireless operation so he would be free to move around on the stage. With the gluiph and an integrated RF unit, he now doesn't even need a "tech table" but can transmit directly to the house system. However, the need for a battery with its considerable weight didn't allow the gluiph to be mounted right next to the trumpet, so an extra belt-mounted box became necessary. On the other hand this will make it easier to handle his other trumpets that we plan to equip with sensors.

3.3.3 Other developments

Currently under construction is a virtual drum that will highlight the possibility for further integration by incorporating the loudspeaker. Its design is based on a subwoofer that has been covered with different materials (leather, velvet, foam pads) to achieve varying deceleration behaviour when hitting it with the accelerometer equipped gloves. The latter also carry small speakers for the higher frequency ranges. This also adds an additional spatial aspect to the instrument apart from the position sensors.

Finally, a DJ mixer is currently being modified to allow for spatial crossfades across a multi-channel output. Other instruments are in the planning phase.

4. THE FUTURE

So where will the gluiph project go from here? While the production of the mainboard seems to be in safe hands with mtronix, the first two projects showed that the amount of customization required is the main factor in determining the amount of time and money involved in designing a gluiph based instrument. With the current one-man workshop situation it is getting increasingly difficult to pursue this.

Therefore it would be desirable to establish an environment where an institution could cover the custom development as well as provide logistic support. Efforts to create such a place in Berlin have proven difficult in the past but carry on.

As far as technology is concerned, the gluiph motherboard will be updated on a regular basis following mtronix' product cycle. A new board with a faster CPU and expanded I/O possibilities is due out for the end of the year. On the software side a port of the recently open-sourced SuperCollider is under consideration.

5. ACKNOWLEDGMENTS

My thanks to Miller Puckette for his initial encouragement and developing pd in general. I'd also like to acknowledge the commitment of mtronix as well as the patience of my first users.

6. REFERENCES

[1] Puckette, M. "Pure Data." Proceedings of the ICMC, 1996.

[2] http://www.steim.org/steim/sensor.html

[3] J.-B. Barriere, P.-F. Baisnee, A. Freed, and M.-D. Baudot "A Digital Signal Multiprocessor and its Musical Application," Proceedings of the ICMC, 1989.

[4] Putnam, W. & Stilson, T. "Frankenstein: A low cost multi-DSP compute engine for the music kit". Proceedings of the ICMC, 1996

[5] Burk, P. "The Integration of Real Time Synthesis into HMSL", Proceedings of the ICMC, 1991.

[6] Curtin, S. "The SoundLab: a wearable computer music instrument", Proceedings of the ICMC, 1994

[7] Avizienis, R., A. Freed, T. Suzuki, and D. Wessel "Scalable Connectivity Processor for Computer Music Performance Systems". Proceedings of the ICMC, 2000.

[8] http://www.la-kitchen.fr/hardw/kroonde.html

[9] Hebel, K. and C. Scaletti. "The Software Architecture of the Kyma System". Proceedings of the ICMC, 1993.

[10] http://www.creamware.de/en/Products/SFP/default.asp

[11] Nicolas Collins, "The Evolution of Trombone-Propelled Electronics," Leonardo Music Journal, Vol. 1, 1991.

[12] http://www.clavia.se/nordmodular/index.htm

[13] http://www.creamware.de/de/products/Noah/default.asp

[14] http://www.plugzilla.com/

[15] http://www.soundart-hot.com/english/index.htm

[16] http://www.gweep.net/~shifty/death/