The OWL Programmable Stage Effects Pedal: Revising the Concept of the Onstage Computer for Live Music Performance

Thomas Webster
University of Bedfordshire
Luton LU1 3JU, UK
thomas.webster@beds.ac.uk

Guillaume LeNost
Lionfish Audio
London SE4 2PB, UK
guillaume@lionfishaudio.com

Martin Klang
Rebel Technology
London E2 8HD, UK
martin@rebel-it.co.uk

ABSTRACT
This paper introduces the OWL stage effects pedal and aims to present the device within the context of Human Computer Interaction (HCI) research.
The OWL is a dedicated, programmable audio device designed to provide an alternative to the use of laptop computers for bespoke audio processing on stage for music performance. By creating a software framework that allows the user to program their own code for the hardware in C++, the OWL project makes it possible to use homemade audio processing on stage without the need for a laptop running a computer music environment such as Pure Data or Supercollider. Moving away from the general-purpose computer to a dedicated audio device means that some of the potential problems and technical complexity of performing with a laptop computer onstage can be avoided, allowing the user to focus more of their attention on the musical performance. Within the format of a traditional guitar 'stomp box', the OWL aims to integrate seamlessly into a guitarist's existing pedal board setup, and in this way presents as an example of a ubiquitous and tangible computing device – a programmable computer designed to fit into an existing mode of musical performance whilst being transparent in use.

Keywords
HCI, embedded computing, Digital Signal Processing, effects pedal, effects patch, Open Source

1. INTRODUCTION
In HCI research, there are a number of areas of discussion related to the design and use of computers in specific contexts, as opposed to relying on general-purpose computers to accomplish particular tasks. Ubiquitous computing relates to the discrete proliferation of computers throughout society – Weiser [14] talks about enhancing computer use by making them available throughout the physical environment, but effectively invisible to the user. This idea could apply to many devices from microwave ovens to digital alarm clocks equally as well as to a variety of digital sound synthesizers and effects units for music performance. Dourish [2] describes tangible computing as an exploration of getting the computer out of the way to "provide people with a much more direct – tangible – interaction experience". Both of these ideas, rooted in HCI research informed the design process of the OWL, as the authors wanted to address a perceived disconnect between performer / instrument and performer / audience that has developed alongside the desire to use bespoke audio programming for music performance and the use of general-purpose computers onstage. By providing a programmable, embedded computing platform for audio processing in the shape of a guitar effects pedal, the project aims to address this issue by making it possible for performers to write their own code and run it on a dedicated and unobtrusive audio hardware device designed around an established music performance paradigm.

2. MOTIVATION
Many gigs now feature performers using laptop computers as musical instruments and sound processors onstage, and there are arguments against this mode of live performance relating to how well the performer is able to interact with the instrument they are playing and the audience observing the performance. Patten et al. [10] note that in the late 1990's, "the transition to laptop performance created a rift between the performer and the audience as there was almost no stage presence to latch onto". This could be ascribed to a combination of the performer having to visually interact with a computer screen, diverting attention, [13], and the limitations of gestural expression afforded by the interface.
Godoy and Leman [4], in a study of musical gestures cite Kurtenbach and Hulteen [8] defining gesture as "A motion of the body that conveys information", and going on to say that "…pressing a key on a keyboard is not a gesture because the motion of the finger on the way to pressing the key is neither observed nor significant. All that matters is which key was pressed". As well as offering a definition of gesture this also highlights the shortcomings of the keyboard as an interface for real-time musical expression in performance. The mouse fares little better in terms of expressive potential; as the main non-discrete means of input, it only allows for movement in two directions, whereas in interaction with traditional musical instruments, performers are capable of a wider variety of more complex gestures.
Drawing a distinction between control and communication gestures in music, Godoy and Leman describe control gestures as conveying information through physical contact (i.e. with a musical instrument), and communication gestures as doing so without physical contact (a nod or motion to a band mate as a cue, perhaps). Later on in the study, when discussing sound-related gestures this category is sub-divided into sound-producing and sound-accompanying gestures, noting that there is some crossover between them – for instance a guitarist may exaggerate a strumming action in a chorus, which could be a combination of: responding to the musical dynamics of fellow performers, intentionally changing the sound from the guitar, and using it as a "theatrical gesture to the audience".

Playing a musical instrument like the guitar also involves physical feedback between the performer and instrument, facilitating a tactile, intuitive and constant refining of the sound being produced. This is a level of physical interaction that is difficult to achieve, and although many laptop performers use external MIDI controllers or re-purpose built-in laptop devices (such as webcams) to add expressive control [3] to the computer, "most of these lack sufficient (visual and) haptic feedback" [1].

Dobrian and Koppelman [1] also discuss playing technique, describing virtuosity as "having complete mastery of an instrument such that s/he can call upon all the capabilities of that instrument with relative ease", and expand upon how this facilitates expression. With less well–established performance instruments and interfaces, there is less potential for virtuosity, as it would be difficult to quickly develop a tradition of virtuoso playing as rich as those of traditional instruments. Designing a computer system to work alongside a traditional instrument rather than replacing it maximizes the expressive potential of experienced players.

It's arguable then that computer interaction for music performance is still unable to match the level of refined gesture and detail of instantaneous expression possible with traditional musical instruments. In light of the ideas presented above, a strong motivation for this project was to design a programmable computing platform for music performance that works alongside a traditional instrument, thereby allowing players to exercise their previously acquired virtuosity in combination with an audio computing device placed firmly in the background, keeping the emphasis on performance and expression.

3. BACKGROUND

It's arguable that any one of a number of digital guitar multi-effects pedals can be examples of ubiquitous and tangible computing devices for the performing guitarist. As a fully programmable device, however, the OWL better fits the expectations of the modern musician as to what should be possible with computing technology. The authors are aware of several existing devices that also aim to place programmable computing power inside dedicated hardware designed for specific musical applications.

The Line6 ToneCore pedal range [1] is comprised of one of a range of interchangeable hardware effect modules that plug in to a base unit containing a Freescale Symphony DSP chip [2], which runs the program from the plug-in module. Various modules are available covering a range of different audio effects. More closely related to the OWL project is the ToneCore developer kit, which has a special base unit with a USB socket and a programmable module. This system allows the user to program an effects patch on an external computer, and upload the code onto the MCU residing in the programmable module. Effects are programmed in assembly language specific to the Freescale Symphony chip, making it difficult to program for many users. The OWL pedal, in contrast, aims to make programming the device more accessible by providing a C++ interface, a more widely used programming language [12].

The OpenStomp guitar pedal [3] a project that began in 2007, is designed around a Propeller Parallax microcontroller [4], can be programmed using a high-level byte coded language called 'Spin' and a low-level assembly language. Some advantages and potential uses of this embedded platform are described by Nagashima [15]. Although Open Source [11] in theory, programming the device requires proprietary software from Parallax, unlike the OWL project which has an Open Source tool chain not dependent on third party activity. Since the development of the OpenStomp project, Parallax has developed a Propeller system that is programmable in C.

Snazzy FX's Ardcore [5] Eurorack [6] format module for modular synthesizer systems is based around an Arduino [7] compatible chipset, which is programmable in C using the Arduino Integrated Development Environment (IDE). The Ardcore is able to do some limited audio processing, but only has an 8-bit converter, which is not high enough resolution for good quality audio applications - the OWL, in comparison has a 24-bit converter better suited for audio processing [8]. The Ardcore module is really only useful for low fidelity audio, or less resolution dependent functions such as controlling the behavior of other synthesizer modules by generating and manipulating Control Voltages [9]. For instance, the module could become a bespoke sequencer, LFO or envelope function for a modular system.

Gonçalves [5], also describes a embedded voltage controlled computer developed for modular systems based on the Arduino platform, presenting the microcontroller as no longer acting as a bridge between the computer and the real world, but instead providing autonomy and computational power for a specific musical system. Amongst future planned work, the author mentions plans for increasing processor speed by upgrading to an ARM series microcontroller, and re-programming the platform in C. These were also concerns for OWL project and as a result the project is implemented around an ARM processor and C++ tool chain.

4. DESIGN & IMPLEMENTATION

When discussing ubiquitous computing, Dourish [2] notes that the difference between a general purpose PC and devices such as microwaves and televisions is that the latter kind of device is organized around human needs and functions and this was a key criterion when designing the OWL. A possible cause of some of the difficulties associated with peripheral computer controllers may lie in the fact that many are designed to fit the performer into the general-purpose computer paradigm, rather than fitting the computer into the performers environment. The authors were therefore conscious of designing a computing platform for a specific purpose, and for it to readily fit in to an existing model of musical performance without it adding complexity to a live situation. In this way the pedal enables guitarists to benefit from the advantages of running their own DSP code, but without compromising the potential energy or interaction of a live performance.

Guitar pedals were considered a good target hardware type for the project as these are an established element of any serious guitarists stage setup [6], and are designed with particular requirements for the sort of situations that they are likely to be used in. The guitarist's 'stomp box' has to have a very high input impedance to carry the signal without degradation from the guitar pickups high impedance output, be sturdy and able to withstand the rigors of stage use, and should be able to be easily switched in and out of a signal chain with a bypass footswitch. In this way, the OWL stage effects pedal is designed with the purpose of integrating into the guitarist's normal

[1] http://line6.com/tcddk/

[2] http://www.freescale.com/webapp/sps/site/homepage.jsp?code=563XXGPDSP&tid=prodlib

[3] http://howleraudio.com/frontpage/

[4] http://www.parallax.com/catalog/microcontrollers/propeller

[5] http://snazzyfx.com/ardcore.html

[6] http://electronicmusic.wikia.com/wiki/Eurorack

[7] http://arduino.cc/

[8] http://hoxtonowl.com/hardware/specification/

[9] http://en.wikipedia.org/wiki/CV/Gate

stage setup and the performance requires no adaptation in order to benefit from its use. Figure 1 shows the final version of the OWL stage effects pedal, with front panel knobs, bi-colour LED pushbutton, and true bypass footswitch.

Figure 1. The OWL stage effects pedal

Amongst many reasons for the rise of the laptop computer onstage is the degree to which the user can tailor musical applications for generating, controlling and processing sound. Having the flexibility to program the pedal in order to configure audio processing for specific requirements was of paramount importance in designing the OWL, as this is something which most digital stage effects pedals are only able to achieve within the parameters of ready-made effects patches.

OWL stands for Open Ware Laboratory, and this acronym reflects an important design consideration in that the hardware and software for the project are Open Source, and freely available under the Gnu General Public License (GPL) [10]. This allows for an iterative design process [13], where users or groups of users in the project community are able to actively contribute to the design and future development of the project by creating new hardware and/or software designs based on the device framework.

Found within mobile phones and other devices, an ARM Cortex (M4) processor provides the computing power inside the pedal, and this is powerful enough for most uses of a stomp box device. Also inside is a 24-bit codec, suitable for high quality AD/DA audio conversion and 1Mb external SRAM. Figure 2 shows the digital circuit board containing the Surface Mount Technology (SMT) components listed above, with header pins for attachment to the counterpart analogue circuit board containing the pots, jacks, pushbutton and footswitch. In this way, modified designs for an alternative analogue circuit board based on this format could be attached, allowing for the development of other types of audio hardware device for specific musical applications.

Figure 2. OWL digital circuit board

[10] http://www.gnu.org/copyleft/gpl.html

The microprocessor can be programmed using a simplified, cross-platform C++ API that allows the user to focus on audio programming without having to worry about the software framework. Effects patches consist of a single .hpp file that inherits simple functions from a parent Patch class. An example of perhaps the most basic patch for the OWL pedal, a volume control, implemented in code is shown in Figure 3.

By providing an API which allows the user to program the device when connected to an external general-purpose computer, bespoke DSP routines can be set up before a gig and modified (if necessary) during a performance via the potentiometers on the surface of the pedal or via a device connected to the expression pedal input. Connection to an external computer is made via micro-USB on the rear of the pedal, and using the USB On-The-Go [11] protocol, it is possible for the device to act as host or peripheral, allowing for the connection of external MIDI controllers.

Patches are written in C++ and complied using an IDE such as XCode or Visual Studio within ready-made projects, or from the Terminal using the Gnu Compiler Collection (GCC). At the time of writing, new patches are loaded by updating the firmware on the pedal, and this can done using the OwlNest GUI application or with a couple of simple commands executed in the Terminal.

5. PATCH LIBRARY

There is already a reasonably sized online patch library [12] consisting of approximately 40 effects patches that users can freely download. Highlights of the library include a reverb based on the work of Jean-Marc Jot [7], ports of effects from the mda [13] plugin collection and the Open Source reverb, Freeverb [14], a phasing algorithm from the Music-DSP Source Code Archive [15], by Ross Bencina [16], and ring modulation, flanging and octave effects from Marek Bareza [17]. Oli Larkin [18] and Charles Verron [19] have provided interesting and original examples of synthesis patches for the OWL platform with their Dronebox and DubSiren patches, illustrating the potential for using the OWL platform in different musical applications. There are also filters & EQ's, compression, overdrive & distortion, modulation patches, delays and many of the standard effects that you would expect to find on a guitarists pedal board. All of these effects patches can be uploaded onto the pedal via a simple GUI interface [20] by a user with no prior programming knowledge; or alternatively could be used as a basis for creating a more complete guitar signal processing chain, involving combinations of several patches.

Due to the Open Source nature of the project, the authors hope to encourage code sharing amongst the user base, and that the patch library will continue to grow as a result. By providing resources for novices wanting to experiment and learn about audio DSP programming in C++, we aim to encourage textual coding and knowledge sharing in audio DSP generally.

[11] http://www.usb.org/developers/onthego/
[12] https://github.com/pingdynasty/OwlPatches
[13] http://mda.smartelectronix.com/
[14] https://ccrma.stanford.edu/~jos/pasp/Freeverb.html
[15] http://www.musicdsp.org/index.php
[16] http://www.rossbencina.com/
[17] http://www.mazbox.com/
[18] http://www.olilarkin.co.uk/
[19] http://www.charlesverron.com/
[20] http://hoxtonowl.com/software/owlnest/

```
#ifndef __GainPatch_h__
#define __GainPatch_h__

#include "StompBox.h"

class GainPatch : public Patch {
public:
  GainPatch(){
    registerParameter(PARAMETER_A, "Gain");
    registerParameter(PARAMETER_B, "");
    registerParameter(PARAMETER_C, "");
    registerParameter(PARAMETER_D, "");
  }
  void processAudio(AudioBuffer &buffer){
    float gain = getParameterValue(PARAMETER_A)*2;
    int size = buffer.getSize();
    for(int ch=0; ch<buffer.getChannels(); ++ch){
      float* buf = buffer.getSamples(ch);
      for(int i=0; i<size; ++i)
    buf[i] = gain*buf[i];
    }
  }
};

#endif // __GainPatch_h__
```

Figure 3. C++ code for a volume control patch

6. SUMMARY & FUTURE WORK

Overall, the authors feel that the OWL project has achieved its goal of offering an alternative to the general-purpose laptop onstage by providing guitarists and performing musicians with a truly programmable computer dedicated to audio processing that fits into an existing mode of live performance.

The project was developed over the course of eight months last year and funded by a Kickstarter crowd-funding [9] campaign [21]. Enough money was raised to make an initial production run of one-hundred and eighty pre-ordered pedals, which shows that the concept of the project has resonated with the target demographic of guitarists with an interest in programming, audio DSP and hardware hacking. We also now have users actively contributing to the online effects patch library, demonstrating not only some of the benefits of an Open Source approach, but also that people are using the device in the way it was intended by developing their own audio DSP code for the pedal.

In addition, the project has developed an Open Source, embedded computing platform that is designed to be modifiable and can be made to fit into a variety of different hardware housings and hacked / repurposed to fit the needs of the user. In this way, other generic uses for an onstage computer such as sequencers and synthesizers could be realized using the same tool chain with similar hardware designs and potentially utilizing more powerful chips such as the ARM A-series.

After evaluating feedback about the project, it's evident that potential users can be discouraged by the idea of programming in C++, and feel the API could be further simplified. The authors agree that there is potential for improvement in this area – either with the development of a GUI / visual programming interface or a simplified IDE for coding patches (in the same vein as Processing or Arduino).

It's been noted that one of the major challenges for people beginning to program in C++ is navigating and learning to use complex IDEs such as XCode and Visual Studio, and finding a solution for this problem to better engage novice programmers is one of the future challenges for the project. Currently we are looking at potential collaborations with existing computer music development platforms such as Faust [22] and Pure Data in order to develop a system that would allow users to create patches within an external computer music environment and then export those patches as C++ code so that they will run on the OWL platform.

[21] https://www.kickstarter.com/projects/marser/owl-programmable-effects-pedal

7. ACKNOWLEDGMENTS

Our thanks to London Music Hackspace [23], AndrewMcPherson [24], Max at sfxsound [25], all at ROLI [26] and Torsten Anders [27].

8. REFERENCES

[1] Dobrian, C. and Koppelman, D. 2006. The 'E' in NIME: musical expression with new computer interfaces. *Proceedings of the 2006 conference on New interfaces for musical expression* (2006), 277–282.

[2] Dourish, P. 2004. *Where the Action is: The Foundations of Embodied Interaction*. MIT Press.

[3] Fiebrink, R. et al. 2007. Don't forget the laptop: using native input capabilities for expressive musical control. *Proceedings of the 7th international conference on New interfaces for musical expression* (2007), 164–167.

[4] Godoy, R.I. and Marc Leman 2010. *Musical Gestures: Sound, Movement, and Meaning*. Routledge.

[5] Goncalves, A. 2011. Towards a Voltage-Controlled Computer Control and Interaction Beyond an Embedded System. *Proceedings of the International Conference on New Interfaces for Musical Expression* (2011), 92–95.

[6] Hunter, D. 2004. *Guitar effects pedals: the practical handbook*. Backbeat.

[7] Jot, J.-M. and Chaigne, A. 1991. Digital delay networks for designing artificial reverberators. *Audio Engineering Society Convention 90* (1991).

[8] Kurtenbach, Gord and Hulteen, Eric 1990. Gestures in Human-Computer Communication. *The Art and Science of Interface Design*. Addison-Wesley Publishing Co.

[9] Ordanini, A. et al. 2011. Crowd-funding: transforming customers into investors through innovative service platforms. *Journal of Service Management*. 22, 4 (2011), 443–470.

[10] Patten, J. et al. 2002. Audiopad: a tag-based interface for musical performance. *Proceedings of the 2002 conference on New interfaces for musical expression* (2002), 1–6.

[11] Perens, B. 1999. The open source definition. *Open sources: voices from the open source revolution*. (1999), 171–85.

[12] Stroustrup, B. 2013. *The C++ programming language*.

[13] Vallis, O. et al. 2010. A shift towards iterative and open-source design for musical interfaces. *Proceedings of the International Conference on New Interfaces for Musical Expression* (2010), 1–6.

[14] Weiser, M. 1993. Some computer science issues in ubiquitous computing. *Communications of the ACM*. 36, 7 (1993), 75–84.

[15] Yoichi Nagashima 2009. Parallel Processing System Design with "Propellor" Processor. *Proceedings of the International Conference on New Interfaces for Musical Expression* (2009), 171-172.

[22] http://faust.grame.fr/
[23] http://musichackspace.org/
[24] http://andrewmcpherson.org/
[25] http://www.sfxsound.co.uk/home/
[26] https://www.roli.com/
[27] https://www.beds.ac.uk/rimap/people/torsten-anders