Pocket Gamelan: realizations of a microtonal composition on a Linux phone using open source music synthesis software

Greg Schiemer[1], Etienne Deleflie[1] and Eva Cheng[2]
[1]Faculty of Creative Arts, University of Wollongong, Wollongong, Australia
[2]School of Electrical Engineering, RMIT University, Melbourne, Australia.
schiemer@uow.edu.au, ed386@uow.edu.au, eva.uow@gmail.com

Abstract. This paper discusses a new approach to computer music synthesis where music is composed specifically for performance using mobile handheld devices. Open source cross-platform computer music synthesis software initially developed for composing on desktop computers has been used to program a Linux phone. Work presented here allows mobile devices to draw on these resources and makes comparisons between the strengths of each program in a mobile phone environment. Motivation is driven by aspirations of the first author who seeks to further develop creative mobile music performance applications first developed in the 1980s using purpose-built hardware and later, using j2me phones. The paper will focus on two different musical implementations of his microtonal composition entitled Butterfly Dekany which was initially implemented in Csound and later programmed using Pure Data. Each implementation represents one of the two programming paradigms that have dominated computer music composition for desktop computers namely, music synthesis using scripting and GUI-based music synthesis. Implementation of the same work using two different open source languages offers a way to understand different approaches to composition as well providing a point of reference for evaluating the performance of mobile hardware.

Keywords: Linux, mobilephone, csound, pure data, chuck, synthesis toolkit.

1 Introduction

The processing capabilities of Linux phones offer the promise of a new playground for music synthesis suitable for the ongoing development of the Pocket Gamelan project in which music for performing ensembles was produced by the first author using j2me phones. Others have already taken the concept of the mobile phone performance ensemble much further initially using Symbian phones and later the iPhone. Open source operating systems and computer music programming resources have facilitated the development of a new genre of interactive musical performance; MoPho performances combine the live synthesis capabilities of RISC processors with new tools for interaction using embedded peripherals [1, 2].

Fig. 1. (a) Neo phone (b) Neo phone development environment

Figure 1 (a) shows the Neo FreeRunner, or Neo, the phone of choice for this project; the phone supports a number of operating systems including OpenMoko and Android and offers similar hardware features as other open source phones. Not only is its software open source, the hardware schematics and CAD specifications for both printed circuit board and molded enclosure are also available open-source.

While a full discussion of the psycho-acoustic, compositional and theatrical reasons for the choice of the Neo are beyond the scope of this paper, the choice was influenced by a need to swing the phone to produce chorusing and Doppler shift which features in works produced to date for the Pocket Gamelan [3]. In short, Neo was chosen because it had a ready-made hole - a significant factor because this eliminated the need to manufacture special pouches for swinging phones.

2 Development environment.

The development environment allows users to create a prototype musical application on a desktop machine and run it on the Neo. It is for users already familiar with making music using one of the open source composition environments ported to the Neo.

It consists of a host machine connected to a client as shown in figure 1 (b). The host is an Acer Netbook running Ubuntu 8.10, and is connected via a USB network cable to the Neo client running Debian. A more detailed description of the preliminary technical work associated with setting up this development environment can be found elsewhere [4].

We modified the Neo to boot two systems: the original OpenMoko 2007.2 distribution (factory installed) from the internal flash memory, or Debian, which we stored on the microSD card; to boot into Debian the auxiliary and power buttons are held and pressed simultaneously.

USB networking on Debian is enabled by default and should always be running. This allows the Neo to communicate via an ssh terminal launched from the Acer. All commands on the Neo must be issued from the Acer terminal.

3 Butterfly Dekany

The work first used to test Csound [5] and Pure Data [6] on the Neo was a microtonal composition entitled Butterfly Dekany composed by the first author. This was an outgrowth of compositions created for j2me phones as part of the Pocket Gamelan project. The work was initially created using Csound for a performance in which a traditional Chinese zither called a Gu Zheng was accompanied by a single battery powered sound source. The first performance was presented at the UNESCO-sponsored GAUNG Music for the New Millenium Workshop in Bedulum, Bali on 29th April 2009. The performer was Eni Agustien.

'Butterfly' in the title refers to a visual characteristic of the Gu Zheng owned by the performer. The instrument has two sets of strings radiating in opposite directions from a central bridge; the bridge forms an axis like the wings of a butterfly radiating from its body on either side. 'Dekany' refers to name of a 10-note microtonal scale devised by contemporary tuning theorist Erv Wilson; this scale is one of a class of scales generated using Wilson's Combination Product Set method [7].

Another microtonal feature is chorusing, a bi-product of the processing algorithm developed for the work as well as the movement of the sound source with its resultant Doppler shift.

One of the features of the work is that sound projection relies on making effective use of the reverberant properties of room acoustics instead of artificial acoustic reinforcement achieved using high levels of electronic amplification. Movement of the sound source and amplitude modulation produced by microtonal intervals and plays an important role in sound propagation.

3.1 Harmonic Organisation.

One of the unique musical properties of scales generated using Wilson's method is that these scales produce groups of harmonically related intervals and chords even though, unlike traditional scales generated from harmonics, the fundamental is not present in the scale. In the case of the scale used for Butterfly Dekany, the set of five harmonics used to generate the combination product set are the harmonics: 1, 3, 7, 9 and 11 which are combined by forming a set of products of pairs of harmonics to produce the scale shown in table 1.

Table 1. Notes of the Dekany formed from product of harmonics 1, 3, 7, 9 and 11

```
; 1:   33/32   undecimal comma                  .3. . . .11
; 2:    9/8    major whole tone               1. . . .9.
; 3:   77/64                                   . . .7. .11
; 4:   21/16   narrow fourth                   .3. .7. .
; 5:   11/8    undecimal semi-augmented fourth 1. . . . .11
; 6:    3/2    perfect fifth                  1.3. . . .
; 7:   99/64                                   . . . .9.11
; 8:   27/16   Pythagorean major sixth         .3. . .9.
; 9:    7/4    harmonic seventh               1. . .7. .
; 10:  63/32   octave - septimal comma         . . .7.9.
```

This scale is in fact two interleaved pentatonic scales, one formed on the odd-numbered notes of the dekany, the other formed on the even notes. Even though the pentatonic scale is one of the most universal scales the harmonic relationship between the two pentatonic scales that occur in this dekany has no cultural precedent.

Butterfly Dekany consists of ten sections in which each section is built on one of the two pentatonic scales. The first - and every subsequent section - is built on notes of the odd-numbered pentatonic scale while the second is built on notes of the even-numbered pentatonic scale. Each half of the Gu Zheng is tuned respectively to the odd- and even-notes of the dekany.

3.2 Scripting paradigm - Csound implementation.

The Csound score for Butterfly Dekany takes the form of layers of sustained notes that enter canonically. Each note consists of five oscillators four of which are pitch-shifted using dynamically changing envelopes in an adaptation of a chorusing algorithm created by Risset [8]. The adaptation involves pitch bends applied to each envelope to form microtonal intervals that occur on the natural harmonic series between harmonics 100, 99 and 98. The microtonal intervals 100/99, 99/98 and 50/49 (i.e. 100/98) are derived from the same set of harmonics used to generate the scale.

Each note produces audible beat fluctuation that slowly accelerates and decelerates over the duration of each note independently of other notes. One of the five oscillators remains on pitch for the duration of the note, while two are pitch shifted above the note at different rates and by different amounts and the remaining two are pitch shifted below the note. Figure 2 shows the Csound instrument created to do this.

```
instr 1
    ibend1  =       1.010101010     ; 100/99
    ibend2  =       1.010204082     ;  99/98
    ipitch  = p5
    itrans  = 0.01
    ifreq1  cps2pch ipitch, -4
    kfreq   =       ifreq

    kover   linseg  0, p3*0.05,  ampdb(p4), p3*0.95, 0, 0.05, 0

    k0      linen kover, 0.01,     p3, p3*0.9
    k1      linen kover, 0.02,     p3, p3*0.8
    k2      linen kover, 0.03,     p3, p3*0.7
    k3      linen kover, 0.04,     p3, p3*0.6
    k4      linen kover, 0.05,     p3, p3*0.5
    k5      linseg ifreq, p3*0.5, (ifreq)*ibend1,      p3*0.4, ifreq
    k6      linseg ifreq, p3*0.4, (ifreq)*ibend2,      p3*0.5, ifreq
    k7      linseg ifreq, p3*0.3, (ifreq)*(2-ibend1),  p3*0.6, ifreq
    k8      linseg ifreq, p3*0.2, (ifreq)*(2-ibend2),  p3*0.7, ifreq

    a0      oscil   k0,  kfreq,   100
    a1      oscil   k1,  k5,      100
    a2      oscil   k2,  k6,      100
    a3      oscil   k3,  k7,      100
    a4      oscil   k4,  k8,      100

    asigl   =       a0 + a1 + a4
    asigr   =       a0 + a2 + a3
            outs    asigl, asigr
            endin
```

Fig.2. The Csound instrument.

Figure 3 shows an extract from the Csound Score.

```
i 1     0       60      68      9.02    i 1     18      42      68      8.02
i 1     3       57      68      9.04    i 1     21      39      68      7.04
i 1     6       54      68      9.06    i 1     33      27      68      8.08
i 1     9       51      68      9.08
i 1     12      48      68      9.10
```

Fig.3. The Csound score.

3.3 GUI-based paradigm - Pure Data implementation.

For the purpose of comparing the relative advantages of the two composition environments the Csound implementation for Butterfly Dekany was rewritten in Pure Data. The canonic entry of voices in Pure Data is implemented using additional instantiations of the instrument where abstractions represent parameter fields in the Csound score file as shown in Figure 4.

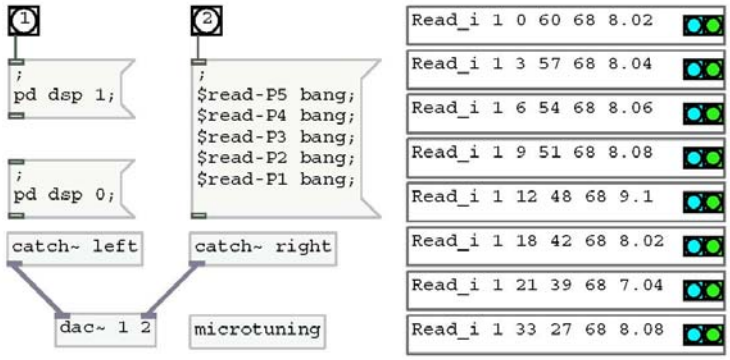

Fig.4. Shows a Pd patch to read the score used in Butterfly Dekany. Each event is arranged in Csound score format with five parameter fields representing instrument number, start time and duration, level (decibels) and scale degree (octave and pitch class). Abstractions in PD correspond to parameter fields shown in Figure 3.

It has proved non-trivial to implement a composition that fitted easily into Csound in a different composition environment. Each line of Csound orchestra code behaves like an object that receives arguments or passes arguments to subsequent lines. Verifying the PD implementation of a process that fits comfortably within Csound sometimes made it necessary to determine the order in which signals are processed or to clarify an underlying concept. The functionality of each line and the flow of information became clearer once it was represented visually in a PD patch and this in turn has opened up new possibilities for live interaction.

One example of this is the left hand column of buttons shown in the Read_i window in Figure 4 above. The left button was introduced as a trigger to test an event out of sequence. This in turn has identified an entry point in the PD implementation where events may be actuated in real time.

Another example is the PD implementation of the envelope generator used to control the overall amplitude of each of the five audio oscillators. Figure 5 shows the PD implementation of the signal called kover which changes dynamically over the duration of each note. The changing output signal is passed to five independent envelope generators. The signal is first packed for transmission before being unpacked sample by sample. This ensures that each of the five envelope generators receive the same sample at the same time.

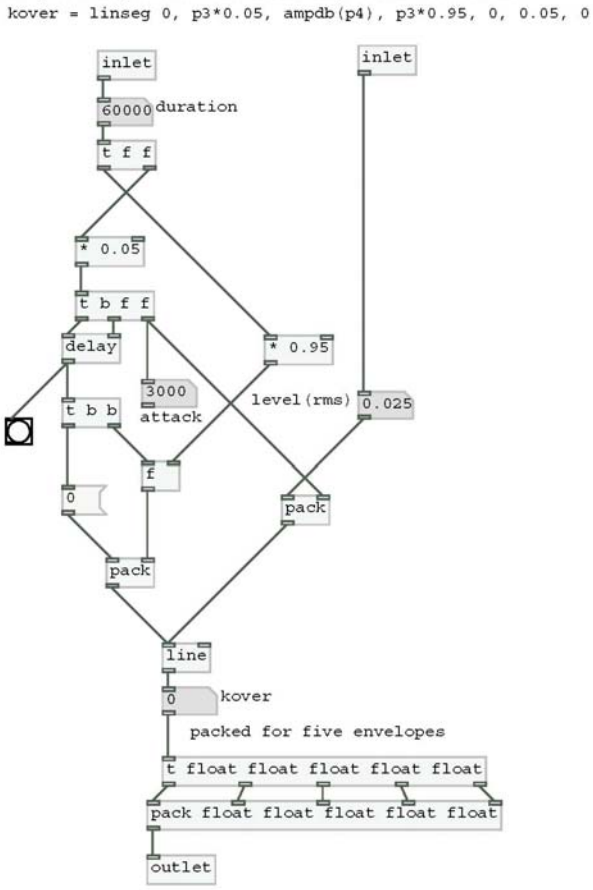

Fig.5. kover – overall envelope control used in Butterfly Dekany.

The transmission of signal from the output of kover to subsequent envelope generators clarified the visual representation of the configuration as represented in the sub-patch shown in Figure 6. This allowed signals read from the parameter fields of the score to be traced more easily on the Read_i parent canvas as shown in Figure 7.

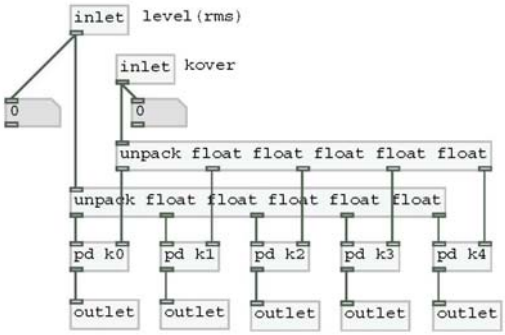

Fig.6. kover distributed to envelope generators k0 to k5.

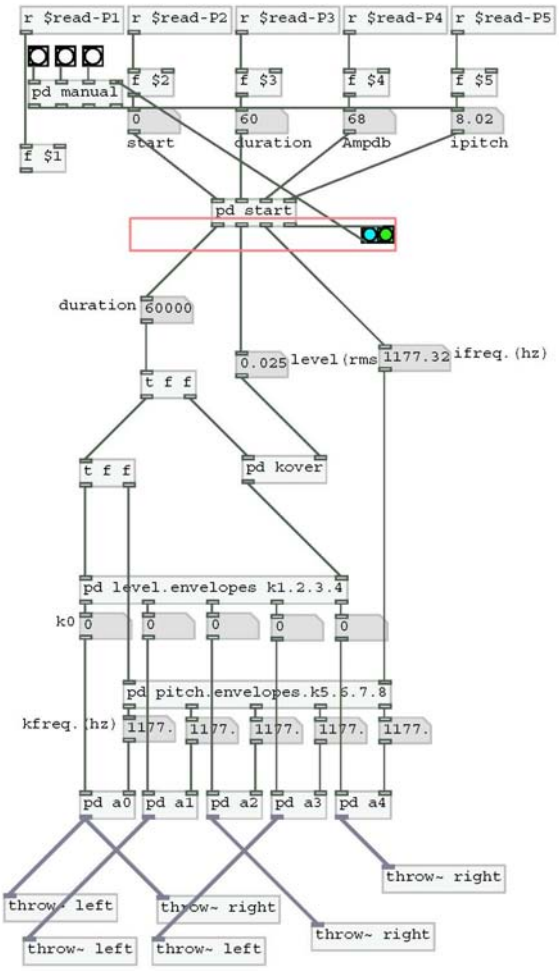

Fig.7. Distribution of duration, level and frequency on the parent canvas.

The parent canvas also illustrates the stereo assignment of outputs of each of the five oscillators; the fixed pitch oscillator is assigned to both channels while the left channel is assigned oscillators with one upper and one lower pitch shift with the remaining two pitch shifted oscillators assigned to the right channel. In a tethered stereo sound system the combination of the speaker assignment and the amplitude modulation produces a sensation of moving sound source.

The implementation of pitch bend is critical for microtonal chorusing. This is represented by the pitch envelope sub-patch k5 shown in Figure 8.

In k5 the microtonal pitch bend is pre-calculated and stored within sub-patch peakbend_A; the value in peakbend_A is the larger microtonal shift of 99/98 above the fixed pitch, while peakbend_a in k6 is a smaller shift of 100/99 above; similarly, peakbend_B in k7 is a larger shift of 99/98 below, while peakbend_b in k8 is a smaller shift of 100/99 below.

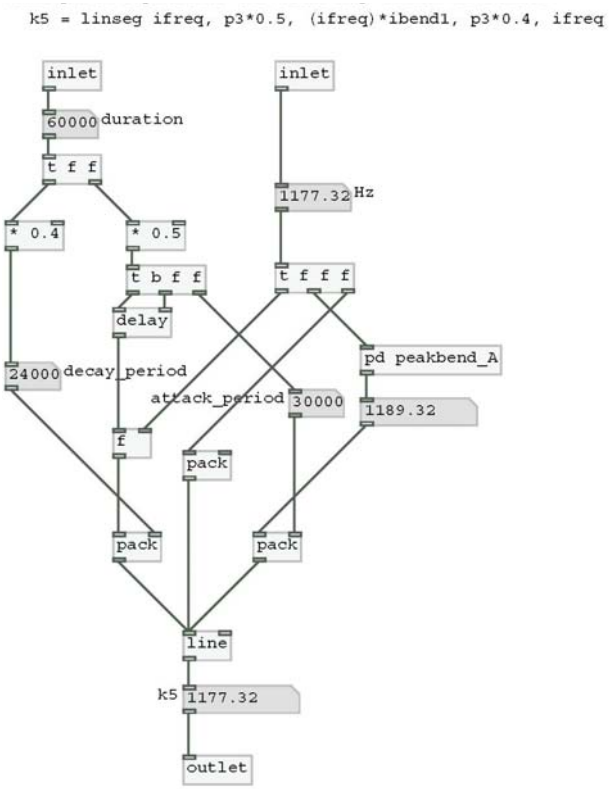

Fig.8**.** Pitch envelope k5.

Having two implementations of one composition allows any drop outs that might occur in the audio pipe to be identified quickly and to form a reliable estimate of the demands an application makes on the mobile hardware. This can be done by observing the output of the jack server jackd which reports dropped frames via the xrun message.

Clearly, the GUI that enabled the application to be prototyped in the desktop environment is no longer needed for launching the application on a mobile device. So the –nogui flag was enabled when the PD implementation of Butterfly Dekany was launched on the Linux phone.

The Csound implementation of the work can also be launched on the Linux phone. And while recent developments in Csound are more suited to real-time interaction than its earlier scripted versions, the real advantage of implementing Butterfly Dekany in PD will be that for future development of similar works there are already many new PD externals that allow real-time control to be refined in mobile devices. Earlier work for java phones by the first author can now not only be emulated in PD on a desktop computer but implemented to run directly on the Linux phone.

4 Conclusion.

Several widely used computer music languages can now be used in conjunction with Linux phones. This brings together the resources of the open source community and a growing body of composers who use these resources for composing music. As these resources migrate to the mobile environment the GUI features of the desktop environment will become less important as other interfaces available in embedded hardware of mobile phones allow us to create instruments that respond to gestural input. We have already done preliminary work and successfully used the Neo hardware with other languages such as STK [9] and ChucK [10]. Plans for the future include a more substantial test of the comparative strengths and weaknesses of various languages using this hardware that would involve creating a single application that can be implemented in each language under test. In addition to a single test based on rendering audio, these tests will need to factor in the effect on processing bandwidth of onboard interface hardware such as accelerometers and wireless network interfaces. In time the Neo or indeed other Linux phones will acquire a community of users who will contribute to the ongoing development of mobile musical applications.

5 Acknowledgements.

Eni Agustien, collaborator in the performance of Butterfly Dekany; Dr Serrano Sianturi and Dr Franki Raden Notosudirdjo, from The Sacred Bridge Foundation, organizers of GAUNG Music for the New Millenium, Bali; Australian Research Council for support for the Pocket Gamelan Project 2003-2005; Mark Havryliv from the University of Wollongong and Norikazu Mitani from IDMI, National University of Singapore for j2me development on the Pocket Gamelan between 2004 and 2008.

References

[1] G. Wang, G. Essl, and H. Pentinnen. “MoPhO: Do Mobile Phones Dream of Electric Orchestras?”. In: Proceedings of the International Computer Music Conference. Belfast, August 2008.
[2] G. Essl and M. Rohs. “Mobile STK for Symbian OS.” In Proceedings of the International Computer Music Conference, New Orleans, Nov. 2006.
[3] Schiemer, G. M. & Havryliv, M. “Pocket gamelan: tuneable trajectories for flying sources in Mandala 3 and Mandala 4”. In: 6th International Conference on New Interfaces for Musical Expression (NIME06) (pp. 37-42). Paris, France, 2006.
[4] Schiemer, G. and Chen, E. "Enabling Musical Applications On A Linux Phone". In: Proceedings of ACMC09, Improvise, The Australasian Computer Music Conference, Queensland University of Technology, 2-4 July 2009.
[5] Boulanger, R. The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing and Programming, MIT Press 2000
[6] Puckette, M. "Pure Data." In: Proceedings, International Computer Music Conference. San Francisco: International Computer Music Association, pp. 269-272. 1996
[7] Wilson, E. (1986) “D'Alessandro Like a Hurricane” in: Xenharmonikôn 9 pp. 1-38.
[8] Risset, J. C. 1969 Introductory Catalogue of Computer-Synthesised Sounds. Bell Telephone Labs., Murray Hill, NJ.
[9] Scavone, G. and Cook, P. “RtMIDI, Rtaudio, and a Synthesis Toolkit (STK) Update" In: Proceedings of the 2005 International Computer Music Conference, Barcelona, Spain 2005
[10] Wang, G. and Cook, P. "ChucK: a programming language for on-the-fly, real-time audio synthesis and multimedia" In: Proceedings of the 12th annual ACM international conference on Multimedia 2004, New York, NY, USA. 2004