Pocket Gamelan: tuneable trajectories for flying sources in *Mandala 3* and *Mandala 4*

Greg Schiemer
Faculty of Creative Arts
University of Wollongong

+61 2 4221 3584
schiemer@uow.edu.au

Mark Havryliv
Faculty of Creative Arts
University of Wollongong

+61 2 4221 3584
mhavryliv@hotmail.com

ABSTRACT

This paper describes two new live performance scenarios for performing music using bluetooth-enabled mobile phones. Interaction between mobile phones via wireless link is a key feature of the performance interface for each scenario. Both scenarios are discussed in the context of two publicly performed works for an ensemble of players in which mobile phone handsets are used both as sound sources and as hand-held controllers. In both works mobile phones are mounted in a specially devised pouch attached to a cord and physically swung to produce audio chorusing. During performance some players swing phones while others operate phones as hand-held controllers. Wireless connectivity enables interaction between flying and hand-held phones. Each work features different bluetooth implementations. In one a dedicated mobile phone acts as a server that interconnects multiple clients, while in the other point to point communication takes place between clients on an ad hoc basis. The paper summarises bluetooth tools designed for live performance realisation and concludes with a comparative evaluation of both scenarios for future implementation of performance by large ensembles of non-expert players performing microtonal music using ubiquitous technology.

Keywords

Java 2 Micro Edition; j2me; Pure Data; PD; Real-Time Media Performance; Just Intonation.

1. INTRODUCTION

In the past decade there has been a paradigm shift from desktop to ubiquitous computing. Mobile phones represent a major part of this shift. Pd2j2me, a set of software tools used to create live musical applications that run in the java 2 micro edition [1, 2], was previously described [3]. It was developed as part of a project that seeks to address the challenge of composing music for mobile computing environments. Called the Pocket Gamelan, the project seeks to develop an interactive musical performance interface that allows non-expert performers to perform microtonal music using mobile phones [3, 4, 5]. In previous papers we described how performance scenarios associated with the project might extend the musical legacy of historical tuning systems as well as new tuning systems first explored by composer and theorist Harry Partch [6] and later extended through the work of contemporary tuning theorist Erv Wilson [7, 8, 9, 10, 11]. Wilson's tuning theories are inspirational because they offer a broad map of the microtonal world that is informed by tuning theorists from many musical traditions. Extensible scales produced by his combination product set theories, and the related Euler-Fokker genus, have a particular appeal for a project focused on the exploration of microtonal music through extensible interfaces.

Two new microtonal works for mobile phones have been created and performed. Microtonal files were created for the Pure Data (Pd) composition environment using recently added features of Scala tuning software designed to export and document tuning data. Purpose-built tools were used to translate Pd files into j2me, a format suitable for java phones. Prior to its realisation using multiple phones, each performance was auditioned using Pd files running on a single desktop machine.

1.1 Pocket Gamelan – pd2j2me

Pd2j2me was implemented as a library of j2me classes that allows composers working in the Pure Data composition environment to create music that can be performed by an ensemble of mobile phones. Since pd2j2me was first developed, two further performance scenarios have been implemented using communication between bluetooth-enabled mobile phone handsets.

A new composition interface has been devised for each work in order to explore the musical possibilities of each scenario. The interface for each application was developed in Pd then exported to the java environment using pd2j2me. Some java was hand coded during the final stages of translation between Pd and j2me. It is our objective, however, that all translation will eventually be automated. The most recent version of pd2j2me was submitted as part of a thesis on 19th December 2005 [12].

2. NEW BLUETOOTH PERFORMANCE SCENARIOS

In performing both the works discussed here, players swing mobile phones on the end of a cord in a circular trajectory, as shown in Figure 1a. Each mobile phone is mounted in a pouch made of semi-transparent fabric attached to a cord.

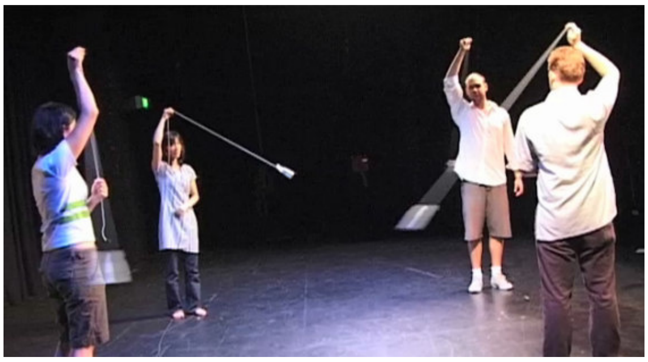


Figure 1a. Mobile phones are used as flying sound sources in *Mandala 3* and *Mandala 4*.

As the phone is swung by its cord it produces audible artefacts such as Doppler shift and chorusing which are produced as a bi-product of movement. The performance concept originated from purpose-built mobile instruments developed by one of the authors more than two decades earlier [13, 14]. One of these instruments, the UFO, is shown in Figure 1b. UFO is an acronym for Ubiquitous Fontana Oscillator, in acknowledgement of sound sculptor Bill Fontana who used flying speakers in 1976.

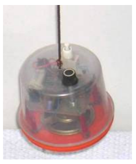


Figure 1b. One of 16 UFOs from the Tupperware Gamelan built for *Mandala 2* in 1981.

Bluetooth communication allows the audio algorithms used in *Mandala 3* and *Mandala 4* to be altered during performance as shown in Figure 1c. During hand-held operation a phone keypad may be operated easily through the fabric as shown in Figure 1d.

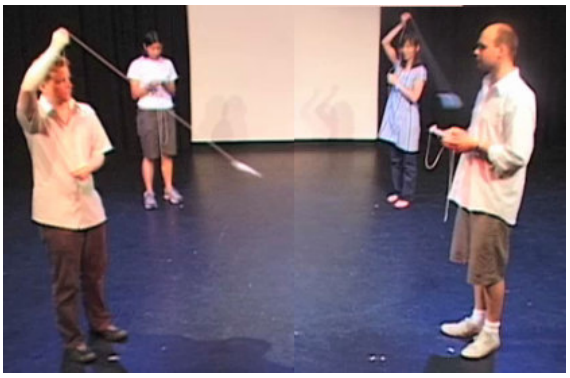


Figure 1c. Flying phones are controlled by hand-held phones.

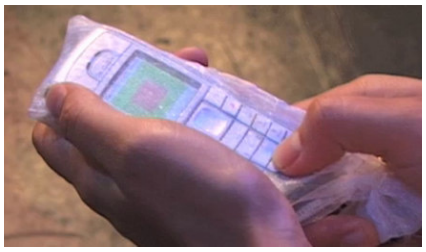


Figure 1d. Pressing a button sends a bluetooth message.

2.1 *Mandala 3*

In the first of these works, entitled *Mandala 3,* three mobile phones (Nokia 6230) interact with one another via a fourth mobile phone (Nokia 7610) which acts as a dedicated server, as shown in Figure 2a. Each 6230 is used both as a sound source and bluetooth controller while the 7610 is used to relay control messages to the other phones. This configuration allows all three sound sources to be affected by the action of any player.

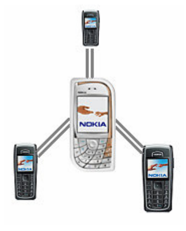


Figure 2a. A mobile 'server' configuration allows any player in the ensemble to affect sound on other phones.

Mandala 3 uses a microtonal scale ascribed to 8th century theorist Al-Farabi [15]. It was first performed at the 13th Australasian Computer Music Conference 2005, Queensland University of Technology, The Loft, Creative Industries Precinct, Kelvin Grove, Brisbane, July 12th 2005.

2.2 *Mandala 4*

In the second of these works, entitled *Mandala 4*, mobile phones communicate with one another on an ad hoc basis. Four Nokia 6230 phones are used both as sound sources and bluetooth controllers; each player in turn operates the phone as a bluetooth controller in order to affect sound on one of the other flying phones. The configuration is shown in Figure 2b.

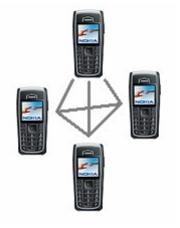


Figure 2b. An ad hoc connection system makes connections only when required.

Each phone is colour-coded as indicated by the colour of its LCD screen. When a single coloured button appears on the screen of a hand-held 6230, a player may choose to respond by sending a control message that affects a flying sound source on another phone. The colour of the on-screen button indicates which phone is affected. The composition specifies a sequence of cues communicated by players during the performance. These cues ensure that each player is ready with phone in hand before the button appears on the screen. As soon as it appears, a player may respond by pressing any key on the 6230. This selects new tuning modes as described in section 4.

Mandala 4 uses one of Wilson's combination product set scales called the Euler-Fokker Genus [15]. This was first performed in the Wild Dog Concert as part of UK Microfest, Riverhouse Barn, Walton-on-Thames, October 15th 2005.

3. DEVELOPMENT ENVIRONMENT – FROM SCALA TO J2ME

Each of the mobile phone performances described above is initially developed in a desktop environment. This allows microtonal aspects of each work to be heard and sequencing of the mobile phones to be simulated. The development path is as follows:

- Scala produces Pd file and exports tuning data to Pd
- Composer works in Pd and simulates sequencing of mobile phones
- pd2j2me produces j2me code
- j2me code is exported to Wireless Toolkit and optionally, refined in Eclipse
- j2me code is compiled and emulated using Wireless Toolkit producing JAR/JAD files
- JAR/JAD files are downloaded to each handset using Nokia PC Suite

The development environment is shown in Figure 3.

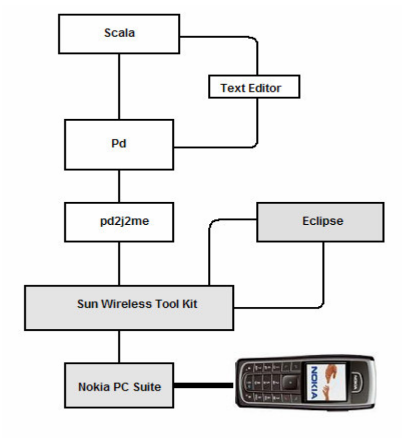


Figure 3. The development path from Scala to deployment to the mobile phone.

3.1 Scala

Scala was chosen for its tuning tools and extensive knowledge base of tuning systems [15]. These include tools to export scales and information related to tuning into other applications. Scala scripting tools extended by Scala's author, Manuel Op de Coul, for the Pocket Gamelan project to allow scale documentation to support file formats used in Pd. Tuning data exported to Pd is stored as a table of linear factors. Linear factors are normally formed by dividing the fraction in a just interval. Scala's tools allow tuning intervals normally measured in cents to be expressed as linear factors. Scales expressed as linear factors simplify tuning operations in Pd because frequencies can be calculated for each scale degree by multiplying a reference frequency by each factor. Extended Scala commands were subsequently released in Scala version 2.2o.

The extended Scala commands gradually evolved into an automated text-based method of generating the Pd interface used in composing these works. As a Pd file was refined or a new canvas added, it was initially saved in Pd. The Pd file was then manually modified using a text editor, and assembled as part of a Scala command file. A new Pd file is then rendered by running the Scala command file. This allowed Pd canvas coordinates and arguments to be initialised consistently, unaffected by the vagaries of the mouse or GUI. In this way Scala managed development of the Pd interface as it grew in complexity. Eventually Pd project management may be automated in Scala using the 'spawn' command.

Scala command files to render Pd applications for *Mandala 3* and *Mandala 4* can be downloaded separately [16].

3.2 j2me

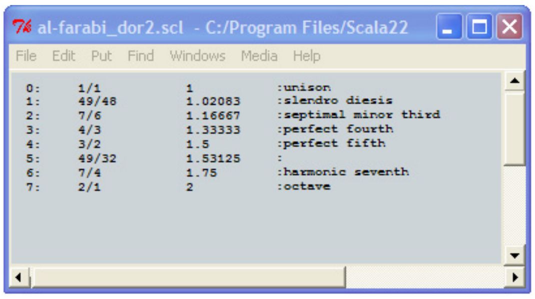
The j2me environment includes:

- pd2j2me – this was discussed previously in 1.1.
- Eclipse – this is used where a particular need is not yet supported by pd2j2me.
- Sun Wireless Toolkit – this compiles j2me source code and simulates the running of the application; it also generates JAR and JAD files which contain the application and application data.
- Nokia PC Suite – this uploads the JAR and JAD files to a target Nokia phone.

4. WORKS

4.1 *Mandala 3*

The Pd canvas in Figure 4a shows the 7-note Dorian scale used in *Mandala 3*.


al-farabi_dor2.scl - C:/Program Files/Scala22
File Edit Put Find Windows Media Help

```
0:    1/1      1        :unison
1:    49/48    1.02083  :slendro diesis
2:    7/6      1.16667  :septimal minor third
3:    4/3      1.33333  :perfect fourth
4:    3/2      1.5      :perfect fifth
5:    49/32    1.53125  :
6:    7/4      1.75     :harmonic seventh
7:    2/1      2        :octave
```

Figure 4a. Al-Farabi's Dorian scale used in *Mandala 3*.

The tuning of Al-Farabi's Dorian scale lends itself to melodic treatment of the kind that can be realized using the Pd interface shown in Figure 4b. This interface plays up to three voices simultaneously. Each voice is a continuous oscillator running on a mobile phone. In the j2me environment each voice is represented by one of three phones, phone A, B and C respectively. Phone functions used in *Mandala 3* will now be explained with reference to the Pd interface shown in Figure 4b.

Single voice melodic phrases are generated on each phone in real-time. Each generator produces a stream of numbers that represent the pitch class of a seven note scale. These address an array containing linear factors used to implement the Al-Farabi tuning.

Pitch classes are generated by using a modulo-7 counter to produce scale steps. Bitwise inversion is then performed on each scale step to vary the melodic contour. Toggle switches in the canvas labelled 'BitwiseInvert1' allow the user to preset various contours. Ascending scales are generated by setting all switches off, descending scales by all switches on and various melodic contours using various combinations of on and off. Pitches may be played over a range of four octaves and start on different scale degrees, or modes, using three groups of horizontal radio buttons labelled 'Keyboard', 'Octave' and 'Mode'. These groups of buttons serve as an input keyboard for users to rehearse with the sequence as well as a status indicator during playback.

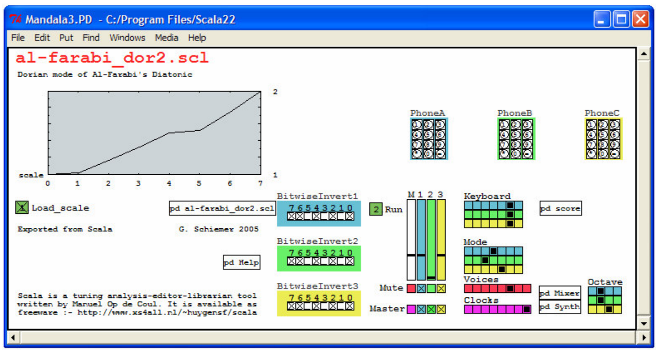

Mandala3.PD - C:/Program Files/Scala22
File Edit Put Find Windows Media Help
al-farabi_dor2.scl
Dorian mode of Al-Farabi's Diatonic
scale
Load_scale
pd al-farabi_dor2.scl
Exported from Scala
G. Schiemer 2005
pd Help
Scala is a tuning analysis-editor-librarian tool written by Manuel Op de Coul. It is available as freeware :- http://www.xs4all.nl/~huygensf/scala
PhoneA
PhoneB
PhoneC
BitwiseInvert1
BitwiseInvert2
BitwiseInvert3
Run
M 1 2 3
Mute
Master
Keyboard
Mode
Voices
Clocks
pd score
pd Mixer
pd Synth
Octave

Figure 4b. Pd interface for *Mandala 3*.

The rhythmic characteristics of melodic phrase are defined by changing the clock period for each note event. The note length of the next event is read from a table of preset values. Events in each phone are sequenced by a local clock. Though each local clock runs asynchronously they are started from the echo server via bluetooth during initialization. There are no discrepancies between the timing of sequences on phones even after a period of twenty minutes. This makes it possible to create elaborate three-voice hockets in which each independent part is rhythmically synchronized.

On the Pd interface, three mute buttons situated below three vertical sliders are used to activate or deactivate each voice. A voice is inactive if its vertical slider is set to zero and its associated mute button is deactivated. The Pd interface also shows two eight-state horizontal radio buttons. The first, labelled ‘Voices’, is also used to activate or deactivate voices; this allows all eight combinations of voices to be selected in a single key operation. The second, labelled ‘Clocks’, allows each sequence to advance or pause independently or in tandem with other voices. Any one-of-eight antiphonal combinations can be selected in a single key operation.

In performance, players select various pre-set combinations by pressing buttons on their phones. This makes it possible for Player A to interact first with Player B and without Player C, then, sometime later, with Player B and with Player C; and so on. On the Pd interface, phone keys are mapped to various control functions using the three twelve-key phone canvases to configure mapping. In performance, the tempo of every phone may be increased or decreased by a factor of 2. This is done by selecting buttons ‘4’ ‘5’ and ‘6’ on any phone; ‘4’ is half speed, ‘5’ is normal speed and ‘6’ is double speed. A tempo change command issued on one handset is relayed by the echo server and takes effect on every handset. This allows any player to select the overall tempo of all sequences. A more complete discussion of mapping for all functions is beyond the scope of this paper.

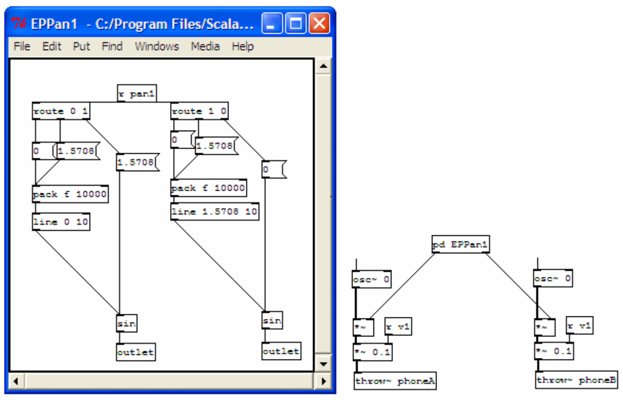
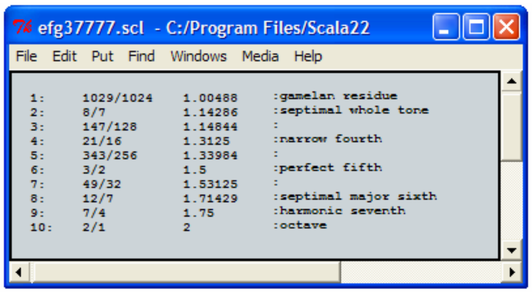


Figure 4c. Pd algorithm allows panning between two phones as shown in sub-patch EPPan1.

The tone generator in each phone also has a second oscillator which doubles at the octave. This allows cross fading between one tone and another an octave apart and results in a gradual timbral change while a sequence is playing. Cross-fading between tones in a single phone is implemented using an equal-powered algorithm based on sine and cosine in the same quadrant. Equal powered cross-fading can also be executed as a form of panning between phones as shown in Figure 4.c. This creates the illusion that sound moves from one phone to another.

4.2 *Mandala 4*

The Pd canvas in Figure 4d shows a 10-note scale used in *Mandala 4*.

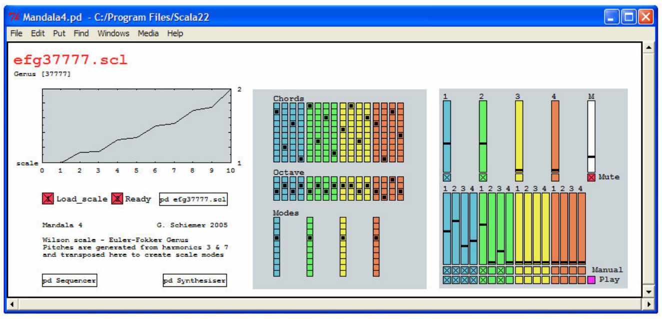

efg37777.scl - C:/Program Files/Scala22
File Edit Put Find Windows Media Help

```
1:    1029/1024   1.00488   :gamelan residue
2:    8/7         1.14286   :septimal whole tone
3:    147/128     1.14844   :
4:    21/16       1.3125    :narrow fourth
5:    343/256     1.33984   :
6:    3/2         1.5       :perfect fifth
7:    49/32       1.53125   :
8:    12/7        1.71429   :septimal major sixth
9:    7/4         1.75      :harmonic seventh
10:   2/1         2         :octave
```

Figure 4d. Erv Wilson’s Euler-Fokker Genus scale based on harmonics 3 and 7 used in *Mandala 4*.

The Pd interface shown in Figure 4e allows control sequences played on all four phones to be programmed and auditioned. Each phone produces four oscillators. Throughout the performance the amplitude of each oscillator is continuously varied using a concatenated series of line functions. The sixteen vertical sliders on the right of Figure 4e allow the composer to monitor changes in amplitude that occur during performance. The composing interface also allows pitch class, octave and transposition to be pre-programmed.

When the performance begins, players assemble in the centre of the stage area to synchronise automated sequences that run throughout the performance. The synchronization sequence takes about 30 seconds and is described in Section 5.2.



Mandala 4 G. Schiemer 2005

Wilson scale - Euler-Fokker Genus
Pitches are generated from harmonics 3 & 7
and transposed here to create scale modes

Figure 4e. Pd interface for *Mandala 4*.

Once all three devices have been synchronised, pre-programmed sequences on all phones begin to play synchronously. Entry and exit of sequences on phones is pre-programmed as shown by the cues in Figure 4f. These sequences ensure that:

- a single cue is given for every event; these provide an aural reference for players during the performance
- each player provides four cues that the other players may use throughout the performance
- every combination of phone solo, duo and trio as well as all four phones is heard.

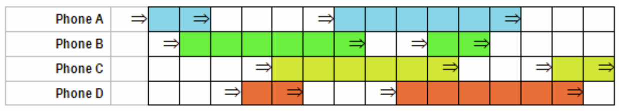


Figure 4f. An automated event sequence determines when sound is produced on each phone. Every 15-30 seconds an event occurs, i.e. a single tone either starts or stops.

In equal temperament, modulation causes pitches to shift to different degrees of the same scale. Here, where just intonation tuning is involved, non-uniform interval sizes cause pitches to shift to positions not audible in the original (i.e. untransposed) scale [16]. In effect, transposition behaves like modulation.

Vertical radio buttons labelled 'Modes' shown in figure 4e, allow intervals of the 10-note scale to be transposed using the same intervals. This is accomplished through the multiplication of linear factors.

In this context, scales played in various transpositions beat with other transpositions of the same scale. This produces beating and other timbral effects which are the bi-products of microtonal intervals produced by modulation.

5. BLUETOOTH TOOLS

While *Mandala 3* and *Mandala 4* both involve wireless communication between several phones, in each case the development process yielded two different ways of managing a bluetooth network: an echo server model implemented as a bluetooth *piconet* and an ad hoc connection system. The development of these was principally driven by a significant limitation in the Nokia 6230's bluetooth implementation: it can only be connected to one other phone at any time.

5.1 Echo server piconet – *Mandala 3*

A bluetooth *piconet* is created when a device is connected to one or more other devices [17]. One device must be the master. In *Mandala 3,* three slaves (6230) communicate bi-directionally with the master (7610), as shown in Figure 2a. Unlike the 6230, the 7610 allows up to seven bluetooth channels to be addressed.

Initially the master device searches for the three phones within a ten metre radius; unsolicited participants are excluded. The phones are then identified with a number – or phone ID – that represents the order in which they are discovered.

Pressing a key on any of the three performing phones generates a message consisting of its phone ID and the key pressed. This message is then echoed by the master to each performing phone. The phone ID and the value of the key pressed determines the response of each phone.

5.2 Ad hoc connection – *Mandala 4*

In *Mandala 4*, the ID of each phone is determined prior to the performance and stored by the phone as its bluetooth friendly name, i.e. the name by which it appears to other devices. Also pre-determined are the times at which one phone attempts to connect to another.

Phones are initially synchronised by Phone A, which searches for the other three phones as soon as the application is run. Phone A then connects and sends a different delay value to each of the other phones which in turn starts a timer. Delay values are pre-computed so all phones begin the piece at the same time.

Once the synchronization sequence is complete, the automated sequences described in section 4.2 begin playing. In the background, each phone then begins a bluetooth discovery sequence searching for other phones taking part in the performance. This allows each phone to identify the bluetooth radio address of other phones to which it must connect during the performance. The bluetooth radio address of each phone discovered is then stored in memory. This allows connections to be made quickly at the appropriate time.

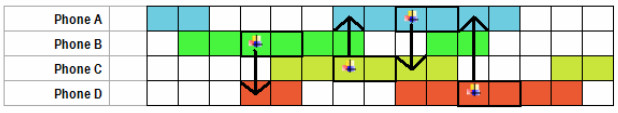


Figure 5. Bluetooth connects a hand-held phone to a flying phone allowing the player to modify its tuning.

Figure 5 shows four windows in the playing sequence during which one phone will automatically connect with another. The player of the phone which initiates the connection is then able to perform operations on a flying phone.

When phone B initiates connection to phone D, as shown by the direction of the arrow in Figure 5, a red button appears on the screen of phone B. This indicates to player B that a new tuning mode may be selected on phone D. Player B may then choose to select a new mode by pressing any key. Mode selection uses the transposition algorithm described in section 4.2. The same thing happens in the remaining three windows shown in Figure 5, where phone C connects to phone A, phone A to phone C and phone D to phone A.

6. CONCLUSION

Work done so far demonstrates the potential of musical applications used with mobile phones. All development has been done using open source software. The tuning resources of Scala, already accessible in many desktop applications like Csound, Artwonk and Metasynth, can now be exported to Pd where pd2j2me is currently used to export applications to java phones. In time, these musical resources can also be extended to other music composition environments like MaxMSP, Supercollider, Algorithmic Composer and AudioMulch, to name a few.

Currently, phones require wired connection for downloading musical applications. In the future, we envisage scenarios in which servers allow tunable applications to be downloaded by multiple clients via a wireless connection. These scenarios provide a broader musical framework in which musicians in future will work with sensor networks based on ultrawideband (UWB) communications [18]. Such networks will dramatically extend the capabilities of bluetooth beyond those explored in *Mandala 3* and *Mandala 4* and address shortcomings of bluetooth implementation in the current generation of phones; we would like all UWB devices to have the capability to multicast in an ad hoc network without the necessity to use an echo server. Moreover, for live musical performance, connection initiated at the discretion of the player is preferable to connection initiated by an automated process; the same is also true of musical games for multiple players [19]. Ongoing development of work begun in *Mandala 3* and *Mandala 4* calls for a new generation of phone that is fully compliant with MIDI, can synthesise streaming audio files and recognise human performance gestures other than the pressing of buttons [20]. This will allow mobile phones to be a generic performance tool widely used by musicians. The frequency hopping algorithm on which mobile phone technology is built has its origins in *Ballet Mecanique*, one of the earliest forays by a twentieth century composer into musical instrument design [21]. Future development of this technology can still be driven by communities of musicians just as such communities drove the development of MIDI more than two decades ago. We see new sensing networks based on UWB potentially as interactive musical instruments that are as diverse as the tuning systems used by musicians over many centuries and in many civilizations.

7. ACKNOWLEDGEMENTS

This project was funded by an Australian Research Council Discovery Grant for 2003-2005. Manuel Op de Coul wrote extensions to Scala source code and offered support during the testing of new scripts.

8. REFERENCES

[1] Sun Microsystems, Java 2 Micro Edition 1999

[2] Connected Limited Device Configuration (CLDC) 1.0, Mobile Information Device Profile (MIDP) 1.0.

[3] Schiemer, G. and Havryliv, M. “Pocket Gamelan: a Pure Data interface for mobile phones” *Proceedings of the New Interfaces for Musical Expression Conference*, *Vancouver, Canada* pp. 156-159 2005

[4] Schiemer, G. and Sabir, K. and Havryliv, M. 2004 “The Pocket Gamelan: A j2me Environment for Just Intonation” *Proceedings of ICMC2004 University of Miami, Florida, November 4th-9th* pp. 654-657 2004

[5] Schiemer, G. Schiemer, G., Alves, W., Taylor, S. and Havryliv, M. “Pocket Gamelan: building the instrumentarium for an extended harmonic universe” *Proceedings of the International Computer Music Conference*, *Singapore* pp. 329 -332 2003

[6] Partch, H. Genesis of Music Da Capo Press 1949

[7] Wilson, E. Musical Instrument US Patent Office Patent Number 3,012,460 1961

[8] Wilson, E. Musical Instrument Keyboard US Patent Office Patent Number 3,342,094 1967

[9] Wilson, E. “The development of Intonational Systems by Extended Linear Mapping” 1975 (1)

[10] Wilson, E. “Bosanquet – A Bridge – A Doorway to Dialog” Xenharmonikôn 3 (13 pages) 1975 (2)

[11] Wilson, E. “D'Alessandro Like a Hurricane” Xenharmonikôn 9 pp. 1-38 1986

[12] Havryliv, M. 2005 “Playing with audio: the relationship between music and games” Master of Creative Arts thesis, University of Wollongong, 2005

[13] Atherton, M. Australian Made - Australian Played University of New South Wales Press, Sydney pp. 210-211 1991

[14] Schiemer, G. “Improvising Machines: Spectral Dance and Token Objects” *Leonardo Music Journal 9* MIT Press pp. 107-114 1999

[15] Op de Coul, M. Scales Archive in Scala available at http://www.xs4all.nl/~huygensf/scala/ 1992

[16] http://www.uow.edu.au/staff/crea/schiemer/.html

[17] Schiemer, G. and Havryliv, M. “Pocket Gamelan: Bluetooth scenarios for mobile phones” *Proceedings of ACMA’2005 Queensland University of Technology, Kelvin Grove, July 12th-15th* pp. 66-71 2005

[18] Nekoogar, F. “Ultra-wideband Communications: Fundamentals and Applications” Prentice Hall, Sydney, 2005

[19] Havryliv, M. and Narushima, T. “Metris: a Game Environment for Music Performance” *Proceedings of Computer Music Modeling and Retrieval, Pisa, Italy* 2005

[20] Schiemer, G. and Havryliv, M. “Wearable Firmware: The Singing Jacket” *Proceedings of ACMA’2004 University of Victoria, Wellington, July 1st-3rd* pp. 66-71 2004

[21] Price, R Further Notes and Anecdotes on Spread-Spectrum Origins see IV. “Shortly before Pearl Harbour: The Lamarr-Antheil Frequency-Hopping Invention” IEEE Transactions on Communications Vol. COM-31, No.1 pp. 89-91 1983