Patterns for the Design of Musical Interaction with Everyday Mobile Devices

Luciano V. Flores[1], Marcelo S. Pimenta[1], Eduardo R. Miranda[2], Eduardo A. A. Radanovitsck[1], Damián Keller[3]

[1]UFRGS – Inst. of Informatics
Caixa Postal 15064, 91501-970
Porto Alegre, RS, Brazil
+55 51 3308 6814
lvflores@inf.ufrgs.br

[2]University of Plymouth
Plymouth, Devon PL4 8AA
United Kingdom
+44 0 1752 586 255
eduardo.miranda@plymouth.ac.uk

[3]UFAC – NAP
Amazonian Center for Music Res.
Caixa Postal 500, 69915-900
Rio Branco, AC, Brazil
dkeller@ccrma.stanford.edu

ABSTRACT
The growing popularity of mobile devices gave birth to a still emergent research field, called Mobile Music (music with mobile devices). Our particular research investigates such repurposing of ordinary mobile devices for use in musical activities. In this paper we propose the use of patterns in the design of musical interaction with these devices. We introduce the musical interaction patterns that came out of our investigation so far, and describe the exploratory prototypes which served as inspiration and, at the same time, as testbed for these proposed interaction patterns.

Keywords
Computer music, mobile music, interaction design patterns, mobile interaction design, mobile devices

INTRODUCTION
The use of computers in music has opened up new possibilities for amateur and professional musicians alike. Research in the field of Computer Music is directed towards the construction of computer systems supporting not only traditional activities (like music composition, performance, music training and education, signal processing and expansion of traditional music sounds, notation study and music analysis), but also some non-conventional and recent activities like music storage and sharing, information retrieval, and classification of musical data (here including music itself and metadata related to music).

Now, combining music and mobile technology promises exciting future developments in a rapidly emerging field, called *Mobile Music* [10]. But this new field is still relatively small. There are only a handful of applications available in a market of mobile device software that is comparatively very large. Although the desktop market for music software is huge, the handheld market has not yet experienced the same level of interest. However, devices such as mobile phones, PDAs, MP3 players, iPhones, and now iPads have already brought music to the ever-changing social and geographic locations of their users, and reshaped their experience of the urban landscape.

Indeed, mobile music technology offers countless new artistic, commercial and socio-cultural opportunities for music creation, listening and sharing. How can we push forward the already successful combination of music and mobile technology? What new forms of interaction with music could be merged into new forms of everyday experiences?

Musical interaction design is the term used in this paper meaning interaction design for systems that support musical activities (either traditional or non-conventional ones). Since we are specifically interested in mobile music, this work suggests the integration of the Computer Music, Human-Computer Interaction, and Ubicomp fields to support musical interaction design for mobile devices.

In the last years, the growing interest from the Computer Music community in digital and computer interfaces for music performance is a clear indication that its researchers have become aware of the importance of Human-Computer Interaction (HCI) studies. Indeed, interfaces for musical performance present immense challenges for HCI: since such interfaces provide the interaction between a performer and a computing system (involving several complex cognitive and motor skills), they make computers turn back toward being things that fit our physical, embodied natures, rather than only operating in the realm of symbolic processing. Therefore, such interfaces for musical performance should be designed around human inputs and outputs rather than computer ones [17], and consequently traditional approaches – which matured and gained their character against a background of office-based personal

IHC 2010 – IX Simpósio de Fatores Humanos em Sistemas Computacionais. October 5-8, 2010, Belo Horizonte, MG, Brazil.
 ISSN 2178-7697

computer applications – are not suitable as design approach.

The need to incorporate HCI theory and practice in the field of musical interfaces has already been pointed out by other authors [13] [25], with primary focus on evaluation, comparison and classification of solutions. However, applying HCI methods for *interaction design* of these solutions or as a way for *improving user experience* remains an open issue.

Clearly, developing musical interaction in mobile devices is a complex and multidisciplinary task with some very interesting challenges. On a high level, these can be divided into two classes:

- Technology-related challenges; and
- Human-related challenges.

Some examples of the first class include studying sensors required in ubicomp, building system software for interoperability and integration, and researching mobile ad hoc networking. As for the latter class, some examples include studying smart home usability, and tools for performing such studies.

We focus on the class of human-related challenges, and emphasize one closely associated with mobile music: *musical interaction design for everyday mobile devices* – i.e. standard, consumer mobile communication and information devices, such as cell phones, smartphones, and PDAs. So, this is the challenge of designing interaction with *non-specific devices*: these devices were originally made for efficient use for other purposes – they were not made specifically to be used in music.

Nonetheless, it is not surprising that most of the work in mobile music focus on the "mobile devices as musical instruments" metaphor. In fact, as Essl states, "it is natural to ask if these devices make good generic platforms for interactive music performance" [8].

In our work, we are *repurposing* those devices for use in musical activities, and as interfaces to musical ubiquitous computing systems, taking benefit from their distinctive capabilities of portability, mobility, and connectivity, and above all from their accessibility to the general public. But before we started to investigate their use in networked, collaborative, or "ubiquitous music" situations, we had to solve the problem of *musical user interaction with the device* itself. Our overall ubiquitous music research is out of the scope of this paper, but the interested reader can find more information in [20].

We have been adopting two main strategies to deal with this platform of non-specific devices and the ubicomp context: (1) designing *interaction* (not interfaces), focusing specifically on how to design musical interaction; and (2) using *interaction design patterns*, which encapsulate interaction design metaphors, good practices and usability attributes, and thus let us free to concentrate on the higher-level objective of design quality.

The goal of this paper is to propose a set of interaction patterns to support the design of musical interaction with everyday mobile devices.

The paper is structured as follows. Next section discusses the use of patterns in interaction design and musical interaction design, presenting motivations, an overview of the main concepts and background on the subject, and related work. Then, we present the four musical interaction patterns that emerged of our investigation so far, and describe the exploratory prototypes which served as inspiration and, at the same time, as testbed for these proposed interaction patterns. The paper ends with a final discussion and concluding remarks.

FROM PATTERNS TO INTERACTION PATTERNS FOR MOBILE MUSIC

One of the major challenges to interaction designers is how to provide detailed guidance and explanations to drive a design in order to achieve quality of use.

High-level design principles are difficult to apply to specific projects, and style guides providing more detailed instructions are often misinterpreted and inaccessible. So, one possible answer is the adoption of *interaction patterns* [2] [24].

Patterns have been applied successfully in software engineering [9]. By definition, a *pattern* is a solution which satisfactorily solves recurring design problems, forming a high-level vocabulary to communicate design issues in a relatively encapsulated way. The concept of patterns in HCI research is fairly new, but relevant for the design community.

The first substantial set of HCI patterns was in the form of a patterns collection, "Common Ground" [3]. Many other collections followed, notably Martijn van Welie's "Interaction Design Patterns" [11], and Jan Borchers' book "A Pattern Approach to Interaction Design" [2]. Some research has been conducted to explore various HCI patterns and to develop *pattern languages* to support user interface (UI) design [4].

In HCI literature we can find at least two kinds of HCI patterns:

- *User interface patterns* or UI design patterns (UIDP), usually related to the "technical" design of various types of user interfaces, such as the design of forms in form-based interfaces, and templates for many widgets used for input/output in distinct applications. They are usually concrete (i.e. platform dependent) patterns, and there exist some UI design pattern libraries [5] [28] and examples of UIDP use [12];
- *Interaction patterns*, usually related to high-level and abstract views for the design of human-computer interfaces. Since they are abstract (platform-independent),

these patterns work for several possible target platforms (desktop, web-based, or even palmtops, cell phones, and iTV-based applications). Examples of interaction patterns may be found in Borchers' book [2].

Interaction patterns are a convenient way to help developers place abstract design rules in the context of concrete projects they are working on. An interaction pattern language can demonstrate how design problems may be solved according to sound user-centered design principles. Indeed, interaction patterns are becoming an important method for bridging the gap between analysis and design in user-centered design. Although several authors of interaction design patterns mention usability as the quality their patterns support, we should not lose track of the global picture: usability is not an end in itself, but a means to reach quality of use.

Interaction Patterns in Mobile Music

Mobile music has recently witnessed a dramatic increase, yet there is still a certain lack of infrastructure to allow broad exploration of what mobile music can really mean. Mobile phones have become attractive platforms for audio and multimedia processing. However, the approach to using them for this end is either based on developing special purpose software, which will offer only one solution, or is based on porting of existing audio and multimedia solutions to mobile phone platforms. Our goal is to offer a generic infrastructure of concepts which can be used by music application designers and developers. We believe that mobile devices are not just small computers, but have inherently different and specific capabilities, and thus require specific resources and approaches for design.

As Beaudouin-Lafon pointed out [1], WIMP interfaces have already reached their limits. These limits are particularly severe in the context of pervasive computing: the amount of information each individual user deals with has grown exponentially; the distribution of this information needs to be deployed over multiple computers and devices, including mainframes, desktop computers, laptops, PDAs, mobile phones and custom hardware; and the range of computer users has expanded drastically, incorporating novices to what was previously regarded as the exclusive realm of experts (music making is a particularly good example). So interaction patterns should inspire metaphors better suited for musical interaction in the context of ubiquitous musical activities.

Based on this assumption, our strategy was to draw on applications, and to develop a set of patterns by reflecting on our experience in developing mobile music applications and music-making applications. These patterns were then organized according to a format inspired in the GammaForm pattern format [9].

In our work on musical interaction with everyday mobile devices we have been able to identify, so far, four interaction patterns which can be implemented in such devices. This identification comes from observing the emergence of common solutions while researching the state of the art in mobile music and exploring existing applications. These four interaction patterns are presented next, and they are being experimented through the construction of some simple interactive application prototypes, so that the patterns may be refined and then used to inform new processes of interaction design for mobile and ubiquitous music.

PATTERNS OF MUSICAL INTERACTION

Our research group has over ten years of expertise in computer music [15]. This has helped us in the process of collecting musical interaction patterns, since we already had experience on the commonly adopted solutions in this domain for interacting with musical data. So, as first step, we did a survey on the state of the art in mobile music applications and, through the lens of our computer music expertise, we could analyze and identify patterns of frequent solutions for musical interaction that were being used in those applications.

In parallel, we did brainstorming sessions to conceive possible musical applications for ordinary mobile devices. We kept as a premise in these sessions, that we should consider many different ways of manipulating music with mobile devices, even if it would require some trade-off between functionality and creative ways of overcoming device limitations. These exercises produced ideas that were tried on the exploratory prototypes which will be described later in this paper. But as a second phase, during prototypes creation, we were already associating the types of musical interaction chosen for each one of the prototypes with the interaction patterns that we were starting to define.

Hence, we began a feedback, exploratory process, in which the emerging patterns were being applied in the design of the prototypes and, in turn, the experience of designing these was being used to refine the patterns.

The Four Proposed Musical Interaction Patterns

General problem statement: All of the four proposed interaction patterns address, in different ways, the general problem of "How may humans *manipulate music* and musical information using everyday (non-specific) mobile devices?" Thus, in a general collection of patterns or a pattern language for mobile interaction design, these proposed patterns could be classified under a "Music Manipulation" or "Multimedia Manipulation" category.

Principles:

- The patterns are musical-activity-independent, i.e. they can support any musical activity, and not just some activity in particular.
- The patterns may be combined to generate more complex designs, as also happens with patterns for other domains (e.g. software design, architecture, etc.).

Pattern: Natural Interaction / Natural Behavior
Solution: Imitate real-world, *natural interaction*.

Description: This pattern corresponds to musical interaction which imitates real interaction with a sound-producing object, or with an acoustic musical instrument. Thus, all musical gestures that we might regard as "natural" may be explored herein: striking, scrubbing, shaking, plucking, bowing, blowing, etc. It relates to the metaphor of "musical instrument manipulation", according to Wanderley and Orio [25], and to the "one-gesture-to-one-acoustic-result" paradigm [26] – hence our alternative label, "natural behavior".

One advantage of designing interaction as a reproduction of *natural musical gesture* is that it will generally include a passive haptic (tactile) feedback, similar to the one we have when interacting with real sound-producing objects. This "primary" feedback (linked to the secondary feedback of hearing the resulting sound) [17] may be important for a "fine-tuned" control of the musical interaction – that "intimate" control suggested by Wessel and Wright [26], which allows the performer to achieve a sonic result that is closer to the intended, and that also facilitates the development of performance technique.

For example, a rhythm performance activity may be implemented using the touchscreen of a PDA, where sounds are triggered when it is gently struck with the stylus, like on a real drum. Or, one may implement a shaker-like instrument by using accelerometer sensors of some mobile device, and musically interacting with this instrument by shaking the device.

But exploring "naturality" in musical interaction design refers not only to designing user input as natural musical gestures, but also to simulating, through UI output, any *natural behavior* which is expected from real-life objects when they produce sound (i.e., behavior that is linked to sound producing phenomena). This can be implemented either through representations on the graphical interface (GUI), or through an adequate mapping, applied to the physical UI, between possible gestures and their naturally expected sonic results.

In our "Drum!" prototype, the user "strikes" the PDA screen and hears a percussion sound, what would be naturally expected. In our "Bouncing Balls" prototype, little "balls" are constantly moving horizontally on the device's screen, making sound every time they "bounce" on "obstacles" (a barrier or the sides of the screen).

Motivation for use: To make musical interaction more "intuitive", that is, to take advantage of what Jef Raskin [21] prefers to call the user's "familiarity" with the interaction. This is justified by the hypothesis that, by designing interaction in a form which "resembles or is identical to something the user has already learned" [21], its learning curve is reduced, what is a usability attribute (learnability).

Pattern: Event Sequencing
Solution: Allow the user to access the timeline of the musical piece, and to "schedule" musical events in this timeline, making it possible for him/her to *arrange a whole set of events* at once.

Description: In this pattern, users interact with music by editing sequences of musical events. This can be applied to any interpretation of these – individual notes, whole samples, modification parameters, in short, any kind of "musical material".

Now, it is important to state that, although our interaction patterns aim primarily at musical control, this does not imply a necessary coupling with performance activities. Neither is this pattern, of event sequencing, useful solely for composition. They are all higher level abstractions which may be applied creatively to any type of musical activity, and should be much more useful if regarded this way. In this sense, it may even be preferable to classify them not under "musical control", but as "music manipulation patterns".

Actually, event sequencing is a good example of this flexibility, since it can be observed both in CODES (asynchronous, compositional tool; see Figure 1) [16] and, for instance, in Yamaha's Tenori-On portable instrument (real-time performance) [18], where the sequences execution is looped, but they can be edited (and so played) in real-time (see Figure 2). This last, synchronous use was also added later to our Drum! prototype (described in the next section), the first prototype in which we combined patterns.

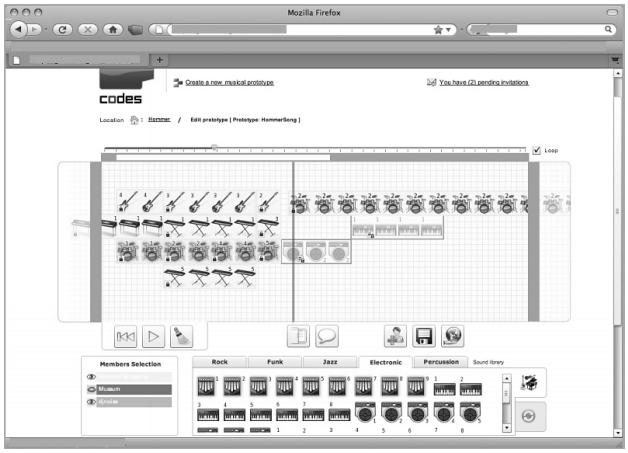


Figure 1. Asynchronous Event Sequencing in CODES, a music composition tool [16].

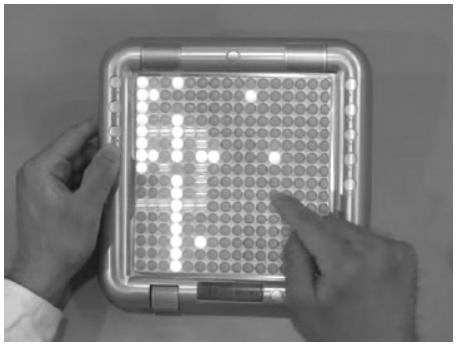


Figure 2. Event Sequencing in Tenori-On, during a looped real-time performance [18].

From designing Drum! and Bouncing Balls we conclude that, by combining interaction patterns, it is possible to create richer interaction.

Motivation for use: Usually, to extend interaction possibilities – increase interaction flexibility – by explicitly allowing, and facilitating, epistemic actions as a complement to pragmatic actions on the system [22] [14].

Pattern: Process Control

Solution: Free the user from event-by-event music manipulation, by allowing him/her to *control a process* which generates musical events or musical material.

Description: This is a well-known interaction pattern in interactive computer music, corresponding to the control of parameters of a generative musical algorithm [27]. It solves that important problem in mobile music, which is the repurposing of non-specific devices: how can we "play" a cell phone, with its very limited keyboard, not ergonomically suited to be played like a piano keyboard?

The Process Control solution suggests a mapping from the (limited) interaction features of mobile devices, not to musical events, but to a small set of *musical process parameters*. This way, we free the user from manipulating music event-after-event, him/her needing only to start the process – which generates a continuous stream of musical events, usually through generative grammars or algorithms – and then to manipulate its parameters. One possible analogy is with the conductor of an orchestra: he doesn't play the actual notes, but he controls the orchestra. This pattern, in fact, corresponds to the "conductor mode" suggested by Dodge and Jerse [6] as one of the possible performance modes in computer music.

For the mapping we find it useful to follow suggestions given by Wessel and Wright [26] when describing their metaphor of a "space of musical processes". Put simple, the idea is that mapping parameters into a key matrix (a keyboard) or a touch-sensitive surface does not need to follow much previous planning: an "intuitive" arrangement of controls in the "parametric space", done by a musician or computer music expert, is enough to yield a satisfactory mapping.

Although it is possible to apply the "parametric navigation" metaphor from these authors, as we did in our Arpeggiator prototype, we believe that it is also possible to use other metaphors they suggest, for the control of interactive musical processes: *drag & drop*, *scrubbing*, *dipping* and *catch & throw* [26]. As for the mapping of control parameters to the different kinds of sensors on mobile devices, we refer to the work of Essl and Rohs [7].

Another useful heuristic for designs using this pattern is that of allowing the user him/herself to configure which process parameters does he/she wants to manipulate.

An example of applying the parametric control of a musical process is the Bloom application for iPhones [19]. This software was developed in collaboration with musician Brian Eno, and allows one to introduce events, through the touch screen, into a generative process. Then, the user may alter the "path" of the process, changing parameters while the music is playing (see Figure 3).

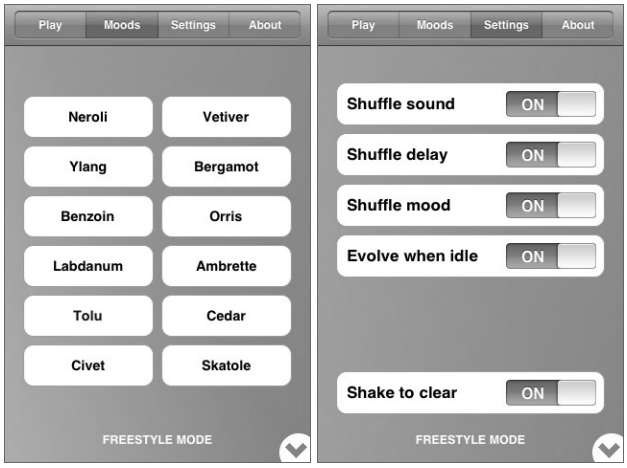


Figure 3. Parameter configuration for a generative musical process in Bloom, an iPhone application [19].

Motivation for use: To avoid the paradigm of event-by-event music manipulation, allowing for more complex musical results through *simpler interaction* with a process, which in turn deals automatically with the details of generating the definitive musical material. This pattern implements HCI principles like *"simplicity"* and *"process automation"*. Since it simplifies interaction, it is also a sound answer to design restrictions imposed by the limitation in interaction features, which is typical of standard mobile devices.

Pattern: Sound Mixing

Solution: Music manipulation through *real-time* control of the parallel execution of longer musical structures (musical material) – i.e. by *mixing* musical material.

Description: This pattern consists in selecting and triggering multiple sounds, so that they may play simultaneously. If a sound is triggered while another is still playing, they are mixed and play together, hence the name of the pattern. Here, music is made as a layered

composition of sounds, but by real-time triggering of events, so we may see sound mixing as the real-time version of event sequencing.

The musical events in this case are sounds or musical structures, and may be of any duration. If they are long (one may even be an entire music sample, triggered just once, or a small but looped sample), we are again avoiding, with this pattern, the traditional note-by-note paradigm of musical control, which is very difficult to implement in conventional mobile devices. But remember: this can be applied not only to music performance. Our "mixDroid" prototype, for example, is a compositional tool where the user records quick, small performances, and combines those into a complete composition.

Sound triggering may be also not necessarily instantaneous. One way to instantiate this pattern is by emulating a real sound mixer (see Figure 4). Sounds will be already playing, but all muted initially. The user will then combine these sounds by manipulating their intensities, maybe gradually. In this form, interaction by sound mixing can be noticed as the method of choice in modern popular electronic music. This form also corresponds to Wessel and Wright's [26] "dipping" metaphor.

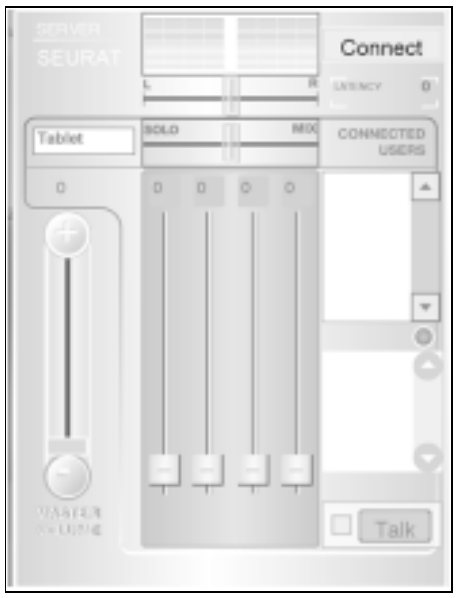


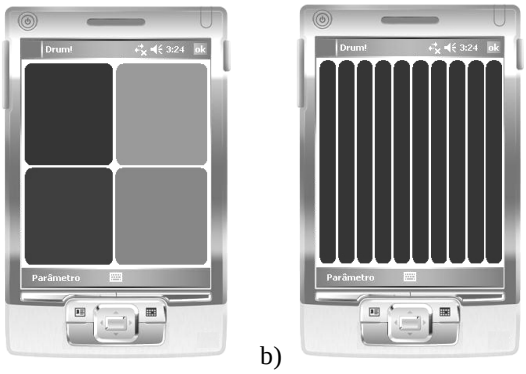
Figure 4. GUI from Tanaka's system for PDAs [23], based on volume-controlled mixing of network transmitted music streams.

Motivation for use: As in Process Control, to avoid the paradigm of event-by-event music manipulation, that is very difficult to implement in conventional mobile devices. Each musical gesture from the user will result in a longer, more complex acoustic result, and the user will be focused in combining these "layers" of sounding musical material.

EXPLORATORY PROTOTYPES

Natural Interaction was explored on our first mobile prototype – Drum! – which came out of the motive of "playing rhythm" and of a brainstorming session in which we manipulated the target device – a PDA. Soon a straight relation was noticed between the device's input modalities and natural musical gestures: we can actually "strike" (gently!) the touch screen with the stylus, like on a real drum. Our first discovery then was that the natural interaction pattern was an elegant way to make effective use of the physical user interface features provided by mobile devices.

We began by creating regions on the screen that would trigger different sounds when "struck" (see Figure 5a). Unfortunately, the "size" of the gesture [26] won't matter in this case. But if we arrange thin triggering regions side by side (see Figure 5b), all playing the same sound, we may use another natural musical gesture – scrubbing – and then eventually explore gesture "size" and a more "intimate" sound-producing control.



a) b)

Figure 5. Two possible implementations of Natural Interaction on the screen of Drum!.

We later added *Event Sequencing* into Drum!, to enrich its musical possibilities. This way, the user may now build a looped background rhythm and improvise over it using the triggering regions. Moreover, the "sequence map" may be edited indirectly, by being set to "record" what is being played with natural gestures (see Figure 6).

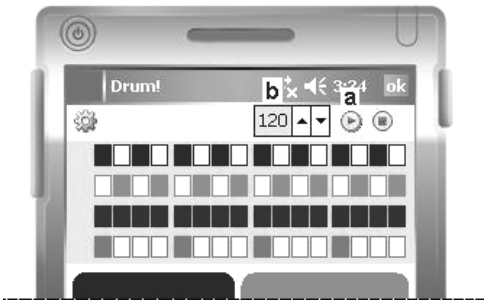


Figure 6. Implementation of Event Sequencing in Drum!.

Bouncing Balls is another rhythmic instrument, which the user plays by choosing the number of balls and their sounds, and then by positioning barriers at 1/4th, 1/3rd or half the way into each ball's horizontal trajectory (see Figure 7). So, this is actually an implementation which also combines interaction patterns, since the input from the user

follows the pattern of *Process Control*, whereas the balls exhibit *Natural Behavior*.

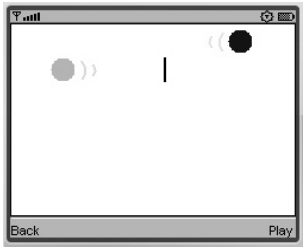


Figure 7. Cell phone screen during a performance with Bouncing Balls, with the lower ball barrier in the middle of the screen.

Our Arpeggiator is an extremely simplified version of a generative musical algorithm, but it is sufficient for our exploration of *Process Control* with mobile devices. The user is freed from controlling the music note by note, needing just to start the arpeggiator process and to control its parameters. In our exploratory prototype, these were mapped to cell phone keys as in rows of a matrix (see Figure 8), so each parameter (*p1*, *p2*, and *p3*) may vary between three possible values.

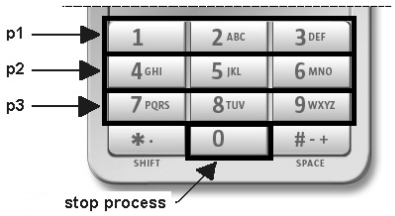


Figure 8. Layout for mapping process parameters into a keyboard matrix in the Arpeggiator.

Control parameters may be pre-defined or user-defined. Our prototype implements one possible combination: *p1* changes arpeggio structure; *p2* changes tonality; *p3* changes tempo; plus, the *0* (zero) key stops the process.

Our mixDroid prototype implements the *Sound Mixing* pattern with buttons that trigger user-assigned sounds (see Figure 9). Although it may be played as a performance instrument (similar as in the first version of Drum!), it was conceived as a composition tool: the user starts a recording, chooses in real-time when each sound will start, and then stops the recording. After, the recorded sequence is reproduced and, again in real-time, it is possible to edit volume and panning of each sound as it plays, which are also recorded. Our intention is to experiment with this synchronous way of composing, which does not rely on a static (usually graphical) representation. The only representation is the composition itself, which is heard in real-time.

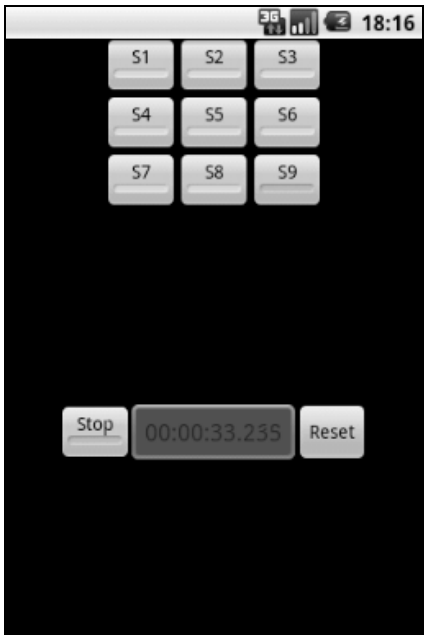


Figure 9. mixDroid's mixing screen.

FINAL DISCUSSION

In our work we identified four musical interaction patterns that can be implemented in common mobile devices. This small, initial set of patterns obviously does not mean to be a thorough taxonomy of musical interaction in general. We are also still on the process of compiling other related pattern sets: for interactions made possible by musical ubiquitous computing environments (i.e., involving cooperation, emergence, location awareness, awareness of contextual sound/music resources, etc.) and for musical interfaces (which instantiate musical interaction patterns, possibly using existing UIDPs). Nevertheless, the four patterns listed here already account for musical interaction in ubiquitous environments when a single mobile device is the user interface, plus they suit designs that need to ensure that music can still be made with a mobile device even with no access to pervasive musical resources (in case those are not available or are unreachable, e.g., due to connectivity limitations).

We have also been conducting preliminary tests on patterns comprehensibility, to observe if the proposed patterns can be learned quickly by designers from outside the CM area. Some other tests are being made to confirm the independence of patterns in relation to different types of musical activities, e.g. by comparing user performance and quality of use when carrying out the same musical activity following two different interaction patterns. These tests and their results will be the subject of forthcoming papers.

A pattern-oriented approach for interaction design in mobile music is an effort towards a necessary switch from the current technology-oriented perspective to a more user-centered perspective of CM as a whole, and this paper is

just a step towards this goal. However, much work is still needed in order to extend the scope of current CM research to cope with many well-known HCI concerns. We are convinced that a better understanding of HCI issues in CM research and development is a good starting point, not only to identify the capabilities and limitations of future work, but mainly to establish a common ground for discussing several interesting questions that are still open.

ACKNOWLEDGMENTS

This work is being partially supported by the Brazilian research funding councils CNPq and CAPES.

REFERENCES

1. Beaudouin-Lafon, M. Designing interaction, not interfaces. In *Proc. AVI '04*, ACM (2004), 15-22.
2. Borchers, J. *A Pattern Approach to Interaction Design*. John Wiley & Sons, Chichester, 2001.
3. Common Ground. http://www.mit.edu/~jtidwell/common_ground.html.
4. Deng, J., Kemp, E., and Todd, E.G. Managing UI Pattern Collections. *ACM International Conference Proceedings Series 94* (2005), 31-38.
5. Design for Mobile. http://patterns.littlespringsdesign.com/.
6. Dodge, C., and Jerse, T.A. *Computer music: synthesis, composition, and performance*. Schirmer Books, 1997.
7. Essl, G., and Rohs, M. Interactivity for Mobile Music-Making. *Organised Sound 14*, 2 (2009), 197-207.
8. Essl, G., Wang, G., and Rohs, M. Developments and Challenges turning Mobile Phones into Generic Music Performance Platforms. In *Proc. MMW 08* (2008).
9. Gamma, E. et al. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1995.
10. Gaye, L. et al. Mobile Music Technology: Report on an Emerging Community. In *Proc. of NIME06*, IRCAM (2006), 22-25.
11. Interaction Design Pattern Library. http://www.welie.com/patterns/.
12. Janeiro, J. et al. Enhancing User Interface Design Pattern Structures. In *Proc. SIGDOC 2009*, ACM (2009), 9-16.
13. Jordà, S. FMOL: Toward User-Friendly, Sophisticated New Musical Instruments. *Computer Music Journal 26*, 3 (2002), 23-39.
14. Kirsh, D., and Maglio, P. On Distinguishing Epistemic from Pragmatic Action. *Cognitive Science 18* (1994), 513-549.
15. LCM – UFRGS Homepage. http://www.inf.ufrgs.br/lcm/.
16. Miletto, E.M. et al. CODES: A Web-based Environment for Cooperative Music Prototyping. *Organised Sound 10*, 3 (2005), 243-253.
17. Miranda, E.R., and Wanderley, M.M. *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. A-R Editions, 2006.
18. Nishibori, Y., and Iwai, T. Tenori-On. In *Proc. NIME06*, IRCAM (2006), 172-175.
19. Opal Limited. Bloom – Generative Music. http://www.generativemusic.com/.
20. Pimenta, M.S. et al. Ubiquitous Music: Concepts and Metaphors. In *Proc. SBCM 2009*, USP/SBC (2009), 139-150.
21. Raskin, J. Intuitive Equals Familiar. *Communications of the ACM 37*, 9 (1994), 17-18.
22. Sharlin, E. et al. On tangible user interfaces, humans and spatiality. *Pers. Ubiquitous Computing 8* (2004), 338-346.
23. Tanaka, A. Mobile Music Making. In *Proc. of NIME'04* (2004), 154-156.
24. Tidwell, J. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media, 2005.
25. Wanderley, M.M., and Orio, N. Evaluation of Input Devices for Musical Expression: Borrowing Tools from HCI. *Computer Music Journal 26*, 3 (2002), 62-76.
26. Wessel, D., and Wright, M. Problems and Prospects for Intimate Musical Control of Computers. *Computer Music Journal 26*, 3 (2002), 11-22.
27. Winkler, T. *Composing Interactive Music*. Cambridge, USA: MIT Press, 2001.
28. Yahoo! Design Pattern Library. http://developer.yahoo.com/ypatterns/.