Beatback: A Real-time Interactive Percussion System for Rhythmic Practise and Exploration

Andrew Hawryshkewich
Simon Fraser University
250 – 13450 102 Ave
Surrey, BC, Canada, V3T 0A3
+1 778.785.0746
aha50@sfu.ca

Philippe Pasquier
Simon Fraser University
250 – 13450 102 Ave
Surrey, BC, Canada, V3T 0A3
+1 778.782.8546
pasquier@sfu.ca

Arne Eigenfeldt
Simon Fraser University
250 – 13450 102 Ave
Surrey, BC, Canada, V3T 0A3
+1 778.782.6786
arne_e@sfu.ca

ABSTRACT

Traditional drum machines and digital drum-kits offer users the ability to practice or perform with a supporting ensemble – such as a bass, guitar and piano – but rarely provide support in the form of an accompanying percussion part. Beatback is a system which develops upon this missing interaction through offering a MIDI enabled drum system which learns and plays in the user's style. In the contexts of rhythmic practise and exploration, Beatback looks at call-response and accompaniment models of interaction to enable new possibilities for rhythmic creativity.

Keywords

Interactive music interface, real-time, percussion, machine learning, Markov models, MIDI.

1. INTRODUCTION

The use of meta-creative or machine learning systems in the imitation or generation of musical material provides possibilities for creating music consistent with a learnt corpus. Combining imitation and generation of musical material leads to a variety of applications, and could include composition of music, augmentation of performance, or simply musical material to explore or interact with. Beatback works by combining imitative and generative systems to create user-focused interactions in percussion exploration, and it is designed with a focus on encouraging rhythmic practise and exploration.

The concept of combining imitative and generative systems is not unique. Systems such as Beatback which employ this combination in a musically reflexive setting are more commonly titled of Interactive Reflexive Musical Systems (IRMS). These can be defined as musical interfaces which enable users to interact with a virtual copy (or mirror) of themselves [1]. Working from this definition, Beatback develops to see if it could offer any benefits within the context of percussion practise and exploration.

The system works by taking user input from standard MIDI drum interfaces – such as trigger pads or a digital drum kit – then learns and generates supporting patterns for their performance. There are two modes of interaction offered – call-response and accompaniment – both of which are modelled off musical performance interactions. In the call-response mode, Beatback simply performs when the user is inactive, enabling a back-and-forth with the system. Whereas in the accompaniment mode, the system fills in drums the user is not playing through use of zoning: Once a drum is struck in a preset region of the kit – such as the toms – none of those drums sound until the user stops striking them (discussed further in Section 3.6). It is these two interaction modes that are being researched with the Beatback system for their approximation of human performer interactions, and their benefits in self-directed practise and exploration.

Within the realm of IRMS, percussion-based systems have not yet been explored extensively, and Beatback is designed to further research this field of interfaces. Looking at how the call-response and accompaniment modes effect users playing on their own provides further insight into how Beatback could benefit solitary practise and exploration of percussion.

2. RELATED WORKS & MOTIVATION

Employing Beatback in a practise setting is meant to help explore the possible applications and benefits in interactive reflexive percussion in practise. Within the field of IRMS, of inspiration to the design of Beatback is the prior work of François Pachet on the Continuator [2], and his discussion of IRMS [1].

The Continuator [1] is a machine learning system which takes a user's input through MIDI enabled controllers (primarily keyboards and guitars), and generates stylistically consistent real-time continuations based on the input. This is all achieved through an application of Variable-Order Markov Models (VOMM) which employ the user's previously recorded input as a learnt corpus from which to generate continuations. Due to the nature of VOMM (discussed in Section 3), these generated continuations tend to be stylistically and rhythmically consistent with the user's input as they maintain some of the musical structure of the inputted patterns, and are generated through a stochastic interpretation of the learnt corpus.

Similarly, Shimon and SHEILA both employ Markov models to store and generate musical patterns based on a user's material. While Shimon is a real-time system, focusing on the use of Markov Models [3], SHEILA instead employs Hidden Markov Models in a non-real-time setting [4]. Shimon demonstrates an effective percussion interaction system, while SHEILA is an example of effective rhythm parsing and generation.

In addition to the generation of continuations, both the Continuator and Shimon work efficiently enough in real-time so that users are able to interact with the system, and in essence, their own musical material. It also provides a form of interaction that moves the focus from just musical expression, to the interaction itself: Rather than simply repeating patterns, those engaging the system would be offered a means to explore their own rhythmic expressions and practise in a reflexive manner. It is this kind of reflexive interaction that Pachet has further explored with Addessi. Of particular interest is the deployment of the Continuator in a children's classroom, which demonstrated that IRMS systems such as the Continuator generated higher levels of intrinsic motivation and focus in children, when compared to its non-IRMS counterpart, the keyboard [5]. Both intrinsic motivation and focus are important in supporting self-directed learning [6], and are key research elements to demonstrating Beatback's strengths.

3. SPECIFICATIONS

Beatback explores different models of self-directed percussion interaction, and therefore requires the ability to generate stylistically appropriate musical material for this interaction. When considering the rhythmic practise and exploration context of Beatback, it is important that the material it generates be consistent with user input in style, and level of complexity. This section details how Beatback works to achieve this: Starting with an overview of its hardware, the focus shifts into a description of how Beatback reads and stores patterns through the use of Variable-Order Markov Models. Then discussed are the means by which pattern generation is triggered, which is outlined along with an overview of drum zoning, and a look at the GUI.

3.1 Overview

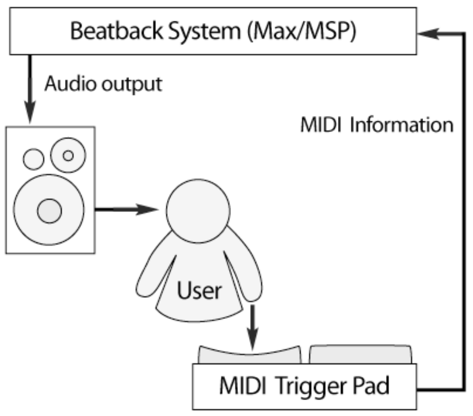


Figure 1: The system setup.

As outlined by Figure 1 and 2, Beatback uses any standardized MIDI interface for input, though it is intended for drum or trigger types of interfaces. The information from the interface is fed into the software end of Beatback – built in Max 5 – which in turn controls audio output for the system. The combination at Beatback's core of MIDI and Max 5 enables the use of a variety of samplers or synthesizers in performance and generation. This also ensures a real-time appropriate latency not available when working with audio signals.

Tempo in Beatback is a fixed value set by the user before engaging with the system (Section 3.8), and can only be changed by adjusting the value according. As parsing and output of patterns is based on the tempo set, material learnt by the system at one tempo can be easily shifted to another.

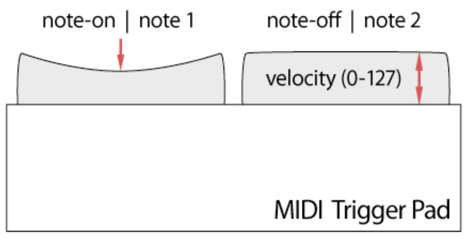


Figure 2: The MIDI trigger pad and input data.

3.2 Parsing MIDI Data

Beatback focuses on percussion, and there are three attributes in particular which it parses from incoming data: Note lengths, velocity and drum type.

Note length values are assigned based upon entry delay between notes, or the temporal distance between two note-on events. Beatback is able to measure these distances based on a tempo set by the user (see Section 3.8), and there are eleven possible note-lengths varying from the maximum of a half-note to a minimum sixty-fourth note (see Table 1 for list). Since note lengths are based on the distance between two notes, there is also an additional value of zero, for when two notes sound together. Velocity is taken in as raw data from the MIDI device and is then quantized into one of eight regions (detailed in Table 1 below). The drum type value is simply a pitch value from the MIDI controller.

Table 1: Quantizing of attribute values.

Attribute	*Quantizing Regions*
Note length	1/2, 3/8, 1/4, 3/16, 1/8, 3/32, 1/16, 3/64, 1/32, 1/64, 0
Velocity	0-15, 16-31, 32-47, 48-63, 64-79, 80-95, 96-111, 112-127
Drums	Up to 12 definable drums (MIDI note values)

3.3 Pattern Input & Learning

As the user plays, the three attributes collected come in as a stream of sequential data. Each note played is read and stored as a combination of the three attributes – drum, velocity, length – and when it comes time for storage, all three are stored together.

Given that the user's input is always being listened for by Beatback, the duration and frequency at which it takes in patterns to be broken down is important. Once every bar, the system will take the patterns input by the user and break them down for storage. In addition to reading once every bar, there is also a maximum of 64 notes played before the system will automatically store the pattern, which is simply to prevent excessively long patterns from being stored in the table. It is also technically unlikely that 64 notes will be played in one bar under normal circumstances.

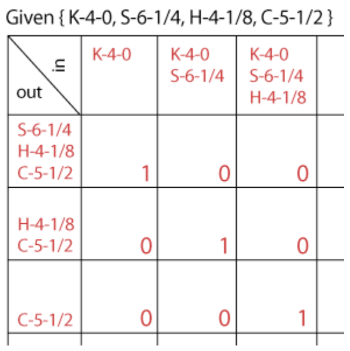
3.3.1 Markov models

In the input and learning of patterns, Beatback uses Variable-Order Markov Models (VOMM) that are probabilistic models in which the state of a process is described by a single discrete variable whose possible values describe all the states of the world [7]. As an example, a common system which functions similarly to how Markov models do is the T9 predictive text system available on most cellphones: As you start to input letters, the system begins narrowing down the possibilities and provides you with the most probable complete word for your set of letters [8]. In the case of Beatback, the complete set being described is the body of patterns being learnt, with the variable being the combined attributes – drum, velocity, length – of the notes. Each set of notes (or pattern) is stored in a table which lists the input and output patterns, and the number of times they have transitioned. The transition probabilities are calculated in the query phase, and are based on the number of transitions that have occurred.

An important distinction between different types of Markov models is the order of the model, which defines how many previous states are considered when computing the probability of future states. In a VOMM, the length of the chain – or in this case pattern – can vary based on the input. This is important musically, as it can consider complete musical patterns of varying lengths, as opposed to fixed-order Markov models, which can only consider a fixed length of input pattern [9]. For example, if given a transition of A to B to C to D, or for this paper's purposes {A,B,C,D}, a first order Markov model will consider on one prior state, and would consider only {B} given {A}, {C} given {B} and so forth. A second order Markov model would result in transitions such as {C} given {A,B}, providing two orders leading to the next value. Beatback draws on the benefit of variable-order Markov models to store all subdivisions of a pattern, while maintaining the original content. Doing so with the previous example would yield transitions such as {B,C,D} given {A}, {C,D} given {A,B} and {D} given {A,B,C}. The pattern itself is not further broken down into smaller portions such as {B,C} given {A} as this begins to remove it further from the musical context of the user's input.

3.3.2 Pattern storage

Presume that the user has input some notes, for which the drum-velocity-length attribute groups are: Kick-60-0 (same time as), Snare-81-1/4. Hi-hat-50-1/8, Crash-68-1/2. Using Table 1 on the prior page to quantize the values, the resulting pattern is {K-4-0, S-6-1/4, H-4-1/8, C-5-1/2}, or {K,S,H,C} for short. Assuming that the bar has just finished and triggered storage, these notes are taken into Beatback inclusion in the pattern table, detailed below.
The system reads the pattern of {K,S,H,C} for storage and separates it into a series of possible in-out combinations which show all the possible starting and ending chains for the pattern. This results in three transitions for the system to store: {S,H,C} given {K}, {H,C} given {K,S} and {C} given {K,S,H}. As show in Figure 3, the drum-velocity-length groupings are stored together in the table, to ensure they are kept associated for generating output. Once stored in the pattern table, the number of times the transition has occurred is updated, in this case to one time each. The calculation of probabilities in Beatback occur


Given { K-4-0, S-6-1/4, H-4-1/8, C-5-1/2 }

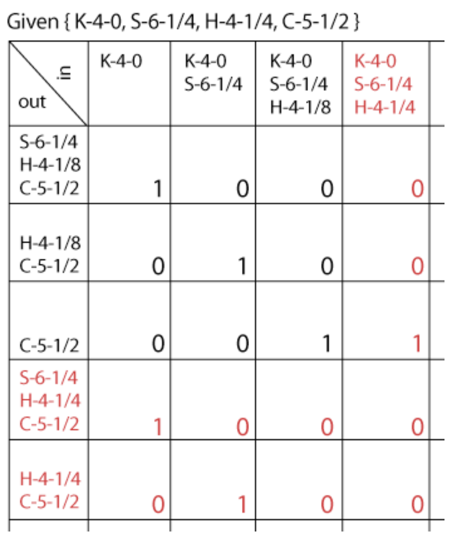
out \ in	K-4-0	K-4-0 S-6-1/4	K-4-0 S-6-1/4 H-4-1/8
S-6-1/4 H-4-1/8 C-5-1/2	1	0	0
H-4-1/8 C-5-1/2	0	1	0
C-5-1/2	0	0	1

Figure 3: First pattern read and stored.

during the querying stage of pattern generation (Section 3.4).
One other important item is how notes that sound together are listed in the pattern. As shown in the example, the kick drum has a length of zero, meaning that it should sound at the same time as the next drum, the snare. When parsing input, Beatback ensures that when two drums sound together, the drum assigned a zero length comes first. The priority for zero-length notes is as follows: Kick, snare, hi-hat, hi-tom, mid-tom, low-tom, crash, and ride.

Now having input one pattern into the table, assume that the next pattern is played by the user and read by the system to be: Kick-49-0, Snare-88-1/4, Hi-hat-58-1/4, Crash-72-1/2. Again referencing the quantization table, this results in a pattern of {K-4-0, S-6-1/4, H-4-1/4, C-5-1/2} or {K,S,H,C} for short. Now the short-form of this pattern is the same as the prior, but the length of the hi-hat note in this pattern is different from the prior. As a result, the table needs to reflect this difference (Figure 4).

Note that even though the drum type pattern was the same, the length of the notes require Beatback to generate new columns and


Given { K-4-0, S-6-1/4, H-4-1/4, C-5-1/2 }

out \ in	K-4-0	K-4-0 S-6-1/4	K-4-0 S-6-1/4 H-4-1/8	K-4-0 S-6-1/4 H-4-1/4
S-6-1/4 H-4-1/8 C-5-1/2	1	0	0	0
H-4-1/8 C-5-1/2	0	1	0	0
C-5-1/2	0	0	1	1
S-6-1/4 H-4-1/4 C-5-1/2	1	0	0	0
H-4-1/4 C-5-1/2	0	1	0	0

Figure 4: Second pattern read is added.

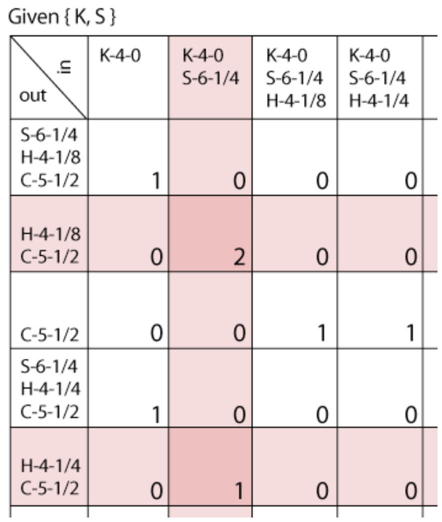
rows to accommodate the new transitions. This ensures that the original rhythms are kept intact even though certain elements of the pattern may be the same. The one exception in this case is with regards to velocity ranges: given two patterns that match in everything except velocity ranges, the transitions will reflect the same pattern being played, though the velocity values will be an average of the preexisting and new velocity value. Having established a pattern table to work with, albeit small, Beatback can now continue and use the learnt information to begin generating patterns.

3.4 Pattern Generation

Pattern generation occurs at the end of every bar. The only point at which Beatback is not generating patterns is when the user first starts the system, as it will not have any learnt material. Once there is material though, Beatback will generate patterns in two stages: Query and build.

In the query stage, the last pattern input (the last bar) by the user is used to query for possible patterns into which to transition. Assume that the last pattern input by the user was Kick-57-1/4 then Snare-84-1/4, {K-4-1/4, S-6-1/4}, shortened to {K,S}. Beatback will try and locate exact matches in the table to see if they exist. Working from Figure 5 (on the next page), there are no exact matches for the full set of drum-velocity-length attributes, so the system removes the velocity attribute and searches again for {K-1/4, S-1/4}. In this case there are still no matches, so Beatback would remove the length attribute, and search for just the drum types {K,S}. As shown in Figure 5, there are two possible matches, so probabilities are calculated. Given {K,S}, there is a 66% chance of transitioning into {H-1/8,C-1/2}, and a 33% chance of transitioning into {H-1/4,C-1/2}, which are then taken into consideration during the build stage.

The build stage uses the probability distributions of possible transitions identified during the query stage and begins to build a pattern. Continuing from the earlier example of {K,S}, one of the possible transitions is stochastically selected from the weighed options, and added to the end of the initial pattern. Let us assume that given {K,S} the less probable transition of {H-1/4,C-1/2} was chosen, which means {K,S,H-1/4,C-1/2} is formed, and the new pattern is output by the system. Beatback then returns this pattern to the query stage only to discover that the new pattern is not in the table. With no listing for {K,S,H-1/4,C-1/2} in the table, the system will start looking again at the sub-patterns off which to build.



Given { K, S }

in / out	K-4-0	K-4-0 S-6-1/4	K-4-0 S-6-1/4 H-4-1/8	K-4-0 S-6-1/4 H-4-1/4
S-6-1/4 H-4-1/8 C-5-1/2	1	0	0	0
H-4-1/8 C-5-1/2	0	2	0	0
C-5-1/2	0	0	1	1
S-6-1/4 H-4-1/4 C-5-1/2	1	0	0	0
H-4-1/4 C-5-1/2	0	1	0	0

Figure 5: After some further input, {K,S} has a transitioned twice into {H-1/8,C-1/2} and once into {H-1/4,C-1/2}

It is important to note that when looking for a transition in the initial query stage, Beatback will take values from the end of the chain if it cannot find entries. In the example, had the system been unable to find entries for {K,S}, it would search for only {K}. Once the initial query stage has passed and a transition has been built onto the original pattern, the removal of values occurs at the beginning of the chain. This is done to prevent the system from always selecting similar transitions as the end of the pattern is the constantly changing element while the system builds the pattern. Resuming the prior example, when {K,S,H-1/4,C-1/2} is not found in the table, Beatback would first remove the length values, and search again for {K,S,H,C}. Search would then repeat with values being removed from the beginning of the pattern until a match is found: None of {S,H,C}, {H,C}, {C} have any transitions associated with them, so at this stage, Beatback would randomly choose a transition from its table to build onto the original pattern.

There are most commonly very few entries near the beginning of the learning process, so random choices will happen most frequently early on. Once the user has input a variety of patterns, the likelihood of having to chose completely randomly diminishes drastically, and output can proceed smoothly.

3.5 Output

As has been discussed, there are two main interaction modes in Beatback: Call-response and accompaniment, which as their names suggest, either respond to inputted patterns or accompany them. In both cases, Beatback begins output after two beats (half a bar) of user inactivity, which is referred to as the output delay. This means that the system's ability to generate material based on the user's input is always two beats behind their performance.

With call-response, once the user has stopped playing and the output delay has transpired, the system will output the generated patterns until the user starts to play again. As soon as the user starts playing again, the system stops.

The output of the system in the accompaniment mode is the exact same as that of the call-response mode, but the output – coupled with the drum-zoning feature (Section 3.6) – filters out the drums that the user is playing. The idea with the accompaniment mode is to support tentative users or to cue possible other rhythmic patterns for the user to explore.

In both modes, the actual output of the system is MIDI data which is sent back to the digital drum-kit's built in sample-bank. Though this data could easily be sent back to a synthesizer, sampler or otherwise.

3.6 Drum Zoning

Perhaps the most significant yet simplest addition to Beatback is the concept of drum zoning. In the zoned system, each drum zone (see Figure 6) can be filtered out by the system when it receives user input within that zone. This enables the system to only fill in drums not being played by the user. While Beatback is preset to use the drum-zoning model listed below, the user can assign their own zoning should they see fit.

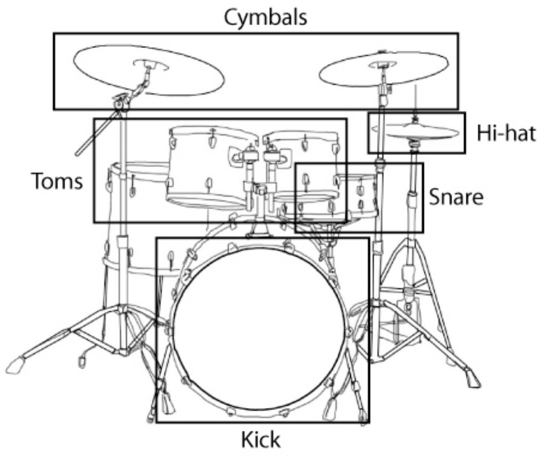


Figure 6: Drum-zones.

Beatback is set up to work with a digital drum-kit, so the model used for zoning it is as follows: The set of cymbals, the toms, the snare, the kick and the hi-hat are all assigned separate zones. As long as the user does not play within a zone, any of the drums within that zone can be played by the system. When the user strikes a drum within a zones – for example the floor tom – none of the drums in that zone – the high, mid and low toms – will sound until the user has not played within that zone for two beats.

The one important contingent for playing an accompaniment in a given zone is that the user has to have played patterns within that zone before. To maintain respect for the users inputted material, Beatback does not take patterns learned with one drum and associate them with another. Therefore to have Beatback play the hi-hat while the user plays the snare, the user first has to play the hi-hat. Then, once the system has learnt how to play the hi-hat, the user can play along with the snare.

Drum zoning offers support to the user when they are playing or learning new patterns. Users could easily load a pattern, and then learn the pattern drum by drum – having the system continue to play the missing parts – until they are comfortable with the entire pattern. Zoning also offers the potential of auditory suggestions in interaction: Beatback could be continuously cueing users with their own musical material on the non-engaged drum zones. This sort of interaction is key to accompaniment which engages and supports the user with their own material while they are performing themselves. The further development of a reflexive yet supportive performance system offers another way of cueing self-directed musical exploration.

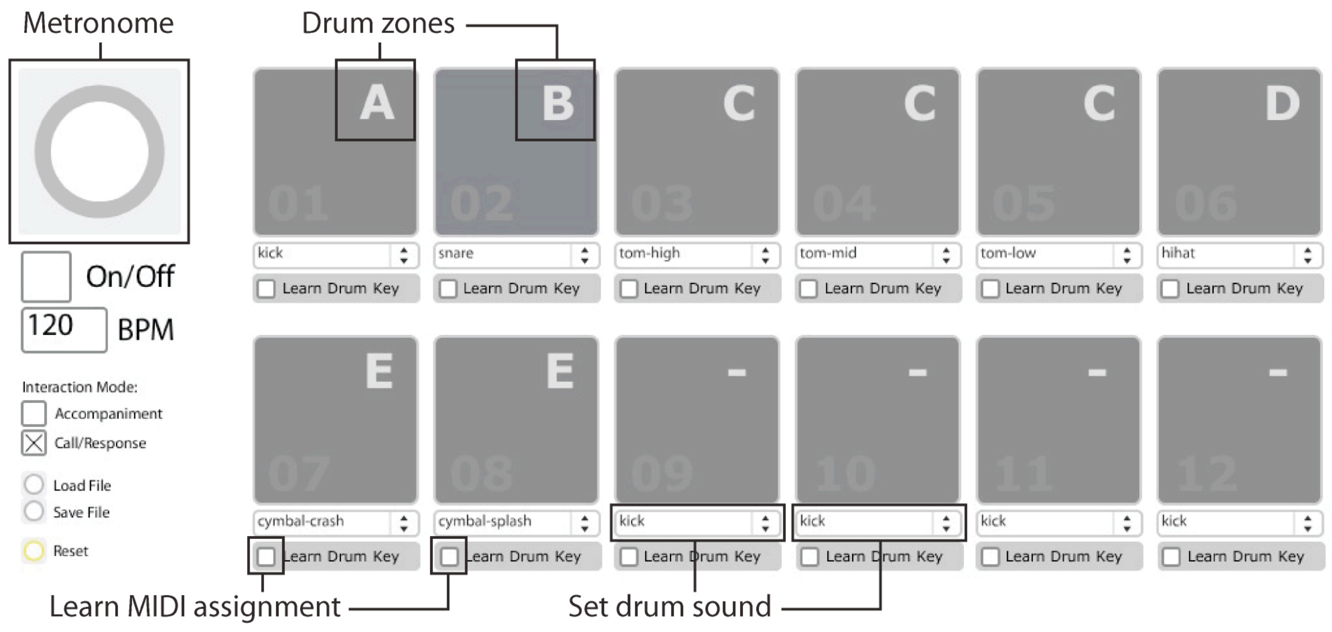


Figure 7: The GUI for Beatback.

3.7 Pattern Storage & Loading

In addition to the ability to learn patterns from user input, Beatback also offers the ability to play along with pre-recorded drum tracks, and to save and load prior sessions. To enable this functionality, Beatback simply encodes and saves the table data – the stored patterns – into a file which can be saved and sent to others. This can be beneficial for users who would like to trade or learn patterns. Learning from a set of drum patterns through zoning enables interactive practise of the pattern, piece by piece.

3.8 The GUI

The GUI for Beatback includes a basic pad-based visual cueing for when a drum pad is struck by the user or the system. A similar visual model using a standard drum-kit is designed to allow for an appropriate visual-to-physical relationship when using a digital drum-kit. In addition to this visual feedback, the GUI allows the user to select different sets of sounds, switch between interaction modes, set tempo, and the zoning of drums.

As seen in Figure 7, there are twelve pads or drums available to the user, a metronome in the top left, and the interaction controls below. Each pad has a switch which enables it to learn a MIDI or keyboard assignment, along with a zone and drum sound setting. In the drum-kit version of the GUI, the zones are preset and the drum-kit will be graphically represented in the GUI itself.

As a final note, users do not engage the GUI during research (Section 4). This is of relevance with regards to IRMS, as Pachet discusses that the lack of GUI interrupting the user makes the Continuator successful as a interactive system [1]: With no other visual stimuli (GUI) present apart from the instrument, the user only has the interface itself to focus on.

4. EMPIRICAL STUDY

As Beatback has been designed for use in rhythmic learning and exploration, the first step in research looks at those who have little to no experience with a drum-kit. In particular, it looks at two different contexts under which they may interact with the system – practice or exploration – and the affordances or benefits offered by the different accompaniment modes. Described below are the results of some early research, and proposed future research. As this paper focuses on detailing the system, this section only provides an overview of results and directions. The underlying interest in furthering this research is to look at how and why Beatback and drum-kit zoning may benefit rhythmic practise and exploration.

4.1 Drum-kit Zoning Research

Under the context of rhythm practise, Beatback offers a system with which users could engage in self-directed learning. Self-directed learning is a model of learning where the student motivates themselves to learn and develop a skill alone [10].

Using this context for learning, the early research looks solely at drum-kit zoning with naive percussionists, or individuals with ten hours or less behind a drum-kit. In the study, participants are introduced to the drum-kit, and given two tasks with which to practise a pattern. For one task, the pattern plays continuously in the background, while for the other task, drum-zoning is applied to their playback. During the task, each participant's performance was recorded as MIDI data, and after each task, they were given a brief – statistically validated [11] – questionnaire on their intrinsic motivation questionnaire to assess perceived enjoyment, tension and competence [12].

Understandably, the majority of participants felt less enjoyment and more tension with drum zoning enabled. Having never played the drums before, many participants felt further overwhelmed when drums would be removed from the pattern as they played them. At the same time, participants felt more confident overall when working with zoning.

The next step in looking at zoning will be with skilled percussionists. In particular, it is expected that the heightened tension and lower enjoyment found in naive percussionists will be reversed for skilled percussionists.

4.2 Future Rhythm Exploration Research

The assessment of exploration of rhythm will focus much more on the use of Beatback in a reflexive musical manner. Using the call-response and the accompaniment modes, this portion of the research will assess how both effect the rhythmic exploration of users.

Similar to the drum-kit zoning research the rhythm exploration research will look self reports of motivation, competence and tension (elements of intrinsic motivation), and the timing accuracy and complexity of the users' performance. Although in the context of timing and complexity, the research in this case will focus more on how the system effects the user's exploration of the two elements. Looking at whether or not the user spends a significant amount of time exploring different types of timing and complexity in rhythm or if they fall into a common pattern will help to understand how a system such as Beatback influences rhythmic exploration.

5. FUTURE WORK

Beatback and it associated research presents an early work which may be beneficial in fostering exploration and engagement in musical creativity. In particular, the assessment of two interaction styles in a different context from the usual self-directed modes could positively inform how to further software based systems for supporting musical creativity or musical learning.

Again, Beatback is only one example of an interactive rhythmic system, and there are a variety of directions for exploring interaction, augmentation and rhythmic creativity. Possible avenues of future development include:

1) *Multi-agent interaction:* Explore using Beatback with more than one user or system simultaneously. It would be of interest to see how interaction occurs between the two users performance while mediated by Beatback.

2) *Develop drum augmentation:* In addition to developing a more full-featured GUI and set of controls, enabling further reading and interpreting of other types of MIDI data such as aftertouch [13] and enhancing velocity sensitivity could offer more control. This could also be of benefit to more professional drummers, looking to augment their own performances.

3) *Explore richer accompaniment:* As Beatback is already capable of being tied to MIDI generated music, research into whether the call-response and accompaniment interaction modes would be enhanced with an ensemble accompaniment would be valuable.

6. CONCLUSION

Beatback demonstrates a system capable of generating live rhythmic responses based on a user's input, and offers a new realm of research into rhythmic exploration and practice. By looking the contexts of rhythmic practice and exploration with the two interaction modes of call-response and accompaniment, Beatback looks to explore how these models may benefit user engaging rhythm. The proposed research of this system will offer further insight into if and how these models of interaction benefit self-directed percussion practice.

7. ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial contributions of the *Natural Sciences and Engineering Research Council of Canada* which have made this paper and research possible.

8. REFERENCES

[1] Pachet, F. Enhancing Individual Creativity with Interactive Musical Reflective Systems. *Musical Creativity: Current Research in Theory and Practice*, Deliege, I. and Wiggins, G, Ed. Psychology Press: London, England, 2006, Section 7.

[2] Pachet, F. The Continuator: Musical Interaction With Style. *Journal of New Music Research*: 32(3), 2003, 333-341.

[3] Weinberg, G., Raman, A. and Mallikarjuna, T. Interactive Jamming with Shimon: A Social Robotic Musician. *In Proceedings of the 4th Human-Robot Interaction Conference (HRI '09),* La Jolla, USA, Mar. 11-13, 2009, 233-234.

[4] Tidemann, A. and Demiris, Y. (2008). A Drum Machine That Learns to Groove. *Proceedings of the 31st annual German conference on Advances in Artificial Intelligence (KI '08),* Heidelberg, Germany, Sept. 23-26, 2008, 144-151.

[5] Addessi, A. R. Interactive Reflexive Musical Systems for Music Education. *In Proceedings of the 1st International Technology, Education and Development Conference (IATED '07)*, Valencia, Spain, Mar. 7-10, 2007.

[6] Fredericks, J. A., Blumenfeld, P. C. and Paris, A. H. School Engagement: Potential for the Concept, State of the Evidence. *Review of Educational Research*: 74(1), 2004, 59-109.

[7] Russell, S. J. and Norvig, P. *Artificial Intelligence: A Modern Approach*. 2nd Edition. Pearson Education, Inc.: New Jersey, USA, 2003.

[8] Dunlop, M. D. And Crossan, A. Predictive text entry methods for mobile phones. *Personal and Ubiquitous Computing*: 4(3), Jun. 2000, 134-143.

[9] Ching, W. K. and Ng, M. K. *Markov Chains: Models, Algorithms and Applications*. Springer Science + Business Media, Inc.: New York, USA, 2006.

[10] Knowles, M. S. (1975). *Self-directed learning: A guide for learners and teachers*. Cambridge Book Company: London, England, 1975.

[11] Tsigilis, N. And Theodosiu, A. Temporal Stability of the Intrinsic Motivation Motivation Inventory. *Perceptual and Motor Skills:* 97, 2003, 271-280.

[12] University of Rochester. *Self-Determination Theory,* Retrieved March 4, 2010, from http://www.psych.rochester.edu/SDT/measures/IMI_description.php

[13] MIDI Manufacturers Association Incorporated. (n.d.). *MIDI Message Table 1,* Retrieved January 17, 2010, from http://www.midi.org/techspecs/midimessages.php