Mark Webber
The Australian National University
Centre for New Media Arts
121 Childers street
Acton, 0200
Australia
U3206695@anu.edu.au

Imitation of the Human Drummer and Beyond: An Advanced Rhythm Generator

Abstract

Drum machines are these days as much a part of popular and contemporary music as the violin or piano were during the classical era. Although these machines have been around for excess of 30 years, they have not really evolved to any great extent and usually involve the often-tedious task of programming steps into patterns, which then need to be sequenced or switched in the correct order. My research has looked at furthering previous drum machine models, turning them into playable instruments for use in live improvisation rather than machines that have to be pre-programmed, pre-prepared and have little or no human interaction

The model described here features GUI and MIDI interface functionality for human control and uses probabilities to generate ongoing variation. Due to this model's functionality it has proven to work extremely well in improvised musical situations and also to successfully mimic the beats and techniques of a human drummer.

Introduction

Drum machines first started appearing on pop music albums during the early 1970s, gaining increasing popularity throughout the decade. By the early 80s many human drummers soon found their skills and talents being replaced or challenged by these rigidly rhythmic machines. Thirty years later various forms of these machines are still an intrinsic part of our current musical culture, not just in pop, but also in other artistically focused genres.

This essay contains two main sections. The first presents a model that uses probabilities within a drum machine's design with the idea of giving the machine a more humanistic functionality, including rhythm, expression and the general sound/feel in comparison to other conventional drum machine models. The aim being to create a drum machine that plays like an instrument and works well when presented with the task of improvising freely with other musicians in an open idea-sharing situation.

The second part provides a technical explanation of how the machine works and insight into how it was developed. This is followed by a consideration of the integration of the drum machine with human musicians and its level of success in working as an inspirational improvisation device.

The use of the term 'drum machine' here in will cover not just the original style hardware machines, but also any music sequencing software and computer programs designed for the creation of drum or rhythmic patterns.

Objectives

Using SuperCollider 3 (SC3) as the basis of my project – "an environment and programming language for real time audio synthesis" (McCartney), my idea was simple: A rhythm sequencer that relies on probabilities to create rhythms, not dissimilar to an 808 style drum machine[1], the difference being that my machine would take a percentage of chance that a sound may get triggered rather than an off/on system as with conventional drum machine design.

The project evolved through an incremental approach to the design, and as my familiarity with SC3 improved the functionality of the drum machine model expanded.

Conventional drum sequencing devices can be generally inferior and less desirable than real drummers in many live musical situations. They lack many of the traits of a well-trained drummer, such as: Feel, groove, variation, chops, dynamics, responsiveness and natural swing. The aim was to create a model capable of being as creative, playable and desirable as a really well trained drummer in a musical performance/improvisation with live musicians.

The Conventional Drum Machine

Drum machines in live performance usually consist of someone setting the tempo, pushing the start button, switching between pre-sequenced patterns while occasionally adding or removing steps and using pre-chosen and pre-compiled drum sounds. This usually results in very little interactivity between performer and machine, and as well as not sounding overly exciting, can also be visually boring. Interaction with other musicians or audience, especially in improvised situations is minimal at best. This perhaps isn't a huge problem in the case of the dj, who is a master of gauging and responding to the audience's mood and also doesn't really apply to bands or groups that are playing pre-rehearsed music in the form of a music concert.

What is more relevant to the context of this model is the musical 'jam': The culmination of musically minded people gathering with the view of creating. In much the same way a painter might approach a canvas without a pre-conceived idea, a group of musicians will

[1] Usually has 16 steps that get cycled through, triggering sounds on the selected beats.

get together and create music. Most musicians know that there are few things more exciting or rewarding as making something out of nothing in an improvisational situation with a group of fellow musicians.

In the case of the drum machine operator there really is not that much creativeness to be applied to what is actually being played. Some may argue that this is not the case and that there are countless different techniques or ways of programming patterns in real time, but in my opinion the operator is still too restricted by and confined to the musical (or non-musical) ideas inherent in the design of the machine.

> The person or corporation that designs the drum machine is often interested in making an instrument that will sell to the greatest numbers of users at the lowest price. The competitive necessity of the corporation to constantly improve the technology ultimately forces the composer and uses of the technology to re-learn and re-accustom themselves to the latest changes. This in-turn greatly influences the composer's original compositional intentions. (Valsamis 2001: pp. 71).

Perhaps a good comparison would be that of the difference between a piano and guitar. The piano player is confined by the workings of the piano. The strings are inside the instrument and can really only be played by hitting a key which in turn swings a hammer against the string. It is a two-step operation. Although the piano can be opened, modified, muted etc. the player is still very much confined by the workings of the instrument. The guitarist, on the other hand, has full access to the strings. They are out in the open to be manipulated freely and easily. The strings can be bent, plucked, hit with mallets, played with a plectrum or fingers. Even the length of the player's fingernails can affect the timbre of the sound produced by the instrument. Although prepared piano pieces use pianos that are modified in various ways, generally they are still a pre-determined performance idea, rather than the result of influence from other players within a musical context. Once the piano is set up in a certain way it is hard to get inside it and apply ideas which occur on a whim, much the same way as with a specifically manufactured drum machine.

The Human Drummer

The points I will be making will assume that by saying 'human drummer' I am actually talking about a properly trained professional or extremely talented drummer with years of training skill and knowledge in the area. Therefore I will be ignoring the simplistic lack-of-experience related problems, such as rhythmic 'tightness', playing on the beat (or unintentionally off as the case may be), varying tempo etc. which are common to the amateur or average drummer, focusing primarily on the restraints of the human body and mind. The first natural and most obvious restraint being that a human drummer only has four limbs and can only hit each drum a restricted number of times per second (which I must admit is an extremely high number of times in the case of professional drummers, who are generally extremely impressive to watch and listen to).

> What advantages are there to using synthetic grooves as opposed to physical, interpersonal grooves in music? One advantage is to transcend limitations of the human body. Epstein suggests that 'biological constraints may exert musical constraints'. Perhaps what Epstein means is that there are direct correlations between physical limitations and the conception of grooves. (Valsamis 2001: pp. 65).

On the other hand, a computer-based drum program is capable of playing a sound so fast that it actually produces a tone. Where a computer falls down in finesse (with drum roles, articulated grace notes etc.), it makes up in raw processing speed and power.

> The physical art and practice of drumming are not requisite to programming a drum machine because playing a drum machine does not require the ability to articulate the sound and rhythms through the body but instead through the mind. (Valsamis 2001: pp. 65).

Secondly, stamina and fatigue can also be problems. A computer can play for hours on end without needing any kind of break or rest. A human on the other hand, gets wary after a few hours straight and would be more likely to make mistakes or not hold the time as well as when they started playing.

Thirdly, it must be considered that a human drummer is just that – a human, influenced by emotional and chemical factors. This isn't necessarily a bad thing, especially when it comes to band and crowd interaction. It can be an advantage, in that a human drummer can feel the vibe and responsiveness of what's going on around them. But performance is also variable due to these factors.

Comparison Between Human and Computer Drumming

The increased popularity of drum machines in pop music during the late 1970s has had an adverse and slightly ironic effect on human pop drummers to date. Early drum machine programmers based their beats upon what the human pop drummers were playing. After a while this swung around so that in fact drum machines were influencing pop drummers into a more rigid way of keeping time, in comparison to drummers of the 1970s and earlier, who would quite happily play behind or in front of the beat, expanding and contracting time. (Valsamis 2001: pp. 70).

During the 1990s programmers became more adventurous with their beats, escaping from the simplistic pop ideas of the 80s and moving into new exciting styles. Highly complex patterns became commonplace among the 'drum and bass' and 'jungle' scene, borrowing ideas and even samples from jazz and funk genres. Many people are aware of the influence of the break beat on many current musical styles. Eventually this aesthetic wore its

way into the pop arena, with groups like *'Everything but the Girl'* and *'Lamb'* finding a middle ground between pop and this new exciting rhythmic style. Eventually human drummers picked up on this style and began to mimic it in their own organic way. Groups such as *'Jungle Funk'* and *'The Bird'* are both good examples of this. Once again the machines had an influence on their human counterparts.

Imitating the Human Drummer

There are still styles yet to be conquered or mimicked successfully by the drum machine. Jazz is at the forefront of these few. So what is it that makes jazz so hard to replicate with a machine? Perhaps it comes down to the massive amount of time put into muscle motor control training required of a jazz drummer and the way this comes across as a finely controlled drumming technique and style. There is so much of the human individual put into this style of playing with many factors contributing to what actually gets produced when the drummer hits the drums. One great factor is the number of different techniques these drummers use to actually play the drums and cymbals. A jazz beat may include a sidestick, a stick hit, a buzz role, and a number of softer ghost notes played on the snare, all in the same short pattern. The jazz drummer is totally engrossed in what the other musicians are playing and will adapt accordingly, catching accents and changing dynamics, allowing the music to move, breath and live. This is very hard to emulate with a machine, and while it's not impossible to create a pattern with different sounds and dynamics to mimic the jazz style, it's near impossible to have the same group interaction; a musical network of shared emotion and ideas.

Groove should also be mentioned at this stage. What is groove and how does one emulate it with a computer? Valsamis quotes Monson's definition of groove: "A rhythmic relation or feeling existing between two or more musical parts and/or individuals." (2001: pp.63). This leads one to think about what this relation or feeling actually is. Andrew McGuiness' recent research into groove music and supporting thesis *Microtiming deviations in Groove* (2004) acknowledges that accent patterns, subtle loudness changes, offset timings, and timbre all contribute to the perception of groove, but the main focus of his research looks at microtiming deviations from quantised onset times. Without delving deeply into this rather broad and slightly scientific topic I will try and present one useful way of incorporating this research within my drum machine.

An extremely brief summary of how McGuiness' model for emulation of groove by a computer works, is that it uses what he calls a 'covert clock', which sets up a cross rhythm to the main clock and pulls the beats towards the time of the covert clock by a specified amount. McGuiness had not fully developed his 'covert clock' theory when I was working on implementing some form of microtiming deviations within my drum machine design, so consequently I have used a different technique based upon his manipulation of data taken from the analysis of the 'funky drummer' break beat. A simple graph of 8 bars shows that the drummer on this recording, taken from James Brown's *In the Jungle Groove* album, consistently deviates slightly behind and in front of the beat over a one bar period. I have taken the lowest and highest values of these deviations over the 8 bars and made the program randomly choose a value close to these points. The level of beat deviation exaggeration can be increased or decreased by the multiplication of these values. Although I recognise that this is not the ultimate way of achieve beat deviations, I believe it to be at least a reasonable starting point.

Beyond Human Drumming

It would seem silly to leave the project at the point of being able to somewhat successfully mimic a human drummer without the exploration of further possibilities, looking into aspects a computer program can accomplish easily, but that are impossible for the human drummer. Changing the pitch of the various instruments is one of my personal favourites. It is nice to have a ride cymbal or snare drum that is not fixed to a particular pitch. Playing the ride cymbal on 32nd notes and moving its pitch around can produce interesting results. I have also added a backwards probability function, meaning that a sample can be given a percentage of chance that it might trigger backwards instead of forwards. Finally, I've taken advantage of the ability to play things super fast and have included a tempo control dial and a BBCut mode (explained in more detail later in the essay). Nick Collins' BBCut classes have the ability to play back samples at a granular level and add excitement to any given drum pattern.

Ignoring Genre and Cliques

> While composers used to define themselves in terms of tonal style (atonality, serialism, octatonic, modal, etc.), those distinctions have been largely superseded by rhythmic content. (Neill 2002, pp. 3).

Conventional drum programming techniques and drum machines make it very easy to fall into the trap of reusing un-original rhythmic ideas. While in pop music this is actually nearly always desired, higher art computer musicians are inevitably looking for new ways to create something different, even if their idea is based upon or inspired by something someone else has produced (which in art is quite often the case). Many current pop-electronica artists are incorporating more experimental rhythmic ideas into their music leading to the creation of new nameless sub-genres, which is resulting in more experimental sounds reaching a broader audience. In the case of my machine, having the ability to create totally random beat patterns allows the composer to avoid imposing the years of sub-conscious genre training upon the beats one creates. It is nice to be able to give the computer a rough guideline of what is wanted and let it make the beat decisions, releasing the composer from the clique temptation.

Neill states that rhythm is the main intersection between popular and high art computer music.

> The two worlds of high art and popular electronic music may use slightly different tools, but their aesthetic approaches are

> most clearly defined in terms of the presence or absence of repetitive beats. (2002, pp. 3).

This thought applies nicely to the aesthetics of my drum machine. Stylistically, the beats produced can be overly popular or highly artistic. The flexibility of the drum machine means it is able to produce anything from the simplest completely repetitive house beat, to a completely rhythmically and musically random piece of computer music.

Technical Details

> Musical computer programs, like any texts, are not "objective" or "universal," but instead represent the particular ideas of their creators. (Lewis 2000: pp. 33).

In brief, sound files containing various drum samples (single hits, not loops) are loaded from a directory into a buffer, which then can be referenced through use of a SynthDef. The program cycles through an array of values that are basically percentages for whether a sound will trigger or not. For example, if the kick drum has a 100% probability value on the first step, then it will be guaranteed to always trigger on that step. If the snare drum has a probability value of 50, on the 3rd step, then it has a 50% chance of being triggered on that step for each cycle.

The point of this is to be able to have a simple pattern that cycles over and over, but is different each time, be it subtly or greatly. Once a sound has been triggered there are a few other choices the program will make. The sound may play forwards or backwards, at any pitch, for any particular length of time (up to the length of the sample), with reverb or another effect possibly added. It may also sub-divide the 16th step into any amount of other beats, meaning it may play the sample 2, 3, or even 24 times within the space of the 16th step/note. The following paragraphs provide a technical explanation of how the program works and the process behind designing it.

Loading Samples

Sound files containing drum samples or other relatively short sounds are first loaded into a buffer using SFLib - an SC3 class built by Newton Armstrong. Files loaded into the buffer are placed and numbered in alphabetical order and can be accessed by reference to their own discreet buffer number. Synth definitions are then loaded which, when called, execute the PlayBuf class:

```
PlayBuf.ar(numChannels, bufnum, rate,
trigger, startPos, loop).
```

Step arrays are created containing probabilities for each instrument ranged between 0 and 100. The length of the step array depends on the set time signature. 4/4 time will create a step array length of 16 probability values (1 bar split into 16 semi-quavers), 6/8 time will create an array of 12 values (1 bar split into 12 semi-quavers) and so forth. Instrument arrays each contain 5 step arrays; 1 for each instrument (ordered kick, snare, hi-hat, ride and other). To make keeping track of each instrument array manageable, they are each given an appropriate name (generally dependant upon style, like '4 to floor' or 'dub beat') and then placed in a master array which can be accessed via the GUI window in a pop-up menu. This allows for probability arrays to be changed in real time while the sequencer is still running, allowing for faster pattern changes and greater versatility.

Main Function

A 'main' function is called using TempoClock class, which is given a tempo in beats per second (bps) and schedules tasks relative to its elapsed beats. I like to work in beats per minute (bpm) rather than bps as I find it easier to know what tempo value I want before I hear it. In order to have 4 semi-quavers per beat this is calculated with: bps = bpm/60 * 4. It is also possible to run two or more TempoClocks in time with each other running on other ratios. At the moment I run two clocks: The standard clock and a second clock, which runs at twice the tempo of the initial clock. The second clock uses 32 length arrays running twice as fast (32nd notes). I've found the second clock to be useful for mimicking a real drummer playing 'grace notes' and I've set the volume of the synths triggered by the second clock to be half that of the initial clock. In the future I would like to run another clock at a different ratio, like 1:3, to create complex polyrhythms, running independent of the main drum beat loop. This would possibly work well for running percussion loops (bongos, tables etc.), in time but cyclically different from the main rhythm.

In effect 'main' gets called 4 times a beat and its primary function is to determine whether a sound is going to be triggered, or not, for each instrument. 'Main' does this by producing a random number and testing it against the current step probability array position to see if the test value is greater than the probability value (obviously it produces a different random number each time it's called). If the test value is less than the probability value, the synth will be called. If it is greater, then it will not get called. For example, if the probability value is 90 and the test value is 94, the sound will not get played. If the probability value is 90 and the test value is 55 the sound will get played. A probability value of 0 has no chance of ever being played, a value of 50 has a 50 percent chance of playing, a value of 100 will always play and so forth. For this reason the step arrays can be seen as basically a list of percentages to determine whether the synth is called.

```
steps1 =
[100,10,10,10,10,100,10,100,10,100,10,1
0,100,10,10,100];
```

Figure 1. *An example of a single step array containing 16 values*

In this example there is a 100% chance that the synth will play on steps 0, 5, 7, 9, 12 and 15. All the other steps have a 10% chance of playing. Using the extreme values (0 and 100) is really important if the aim is to create a somewhat repetitive rhythm. Placing a 100% value on the first beat of the pattern for, say the kick, gives the pattern a feeling of cyclic rhythm. Many musical styles rely heavily upon the listener and other

musicians being able to identify the 1st beat and thus the start of each pattern cycle. Setting every single probability to 50% would take away the 4/4 meter and feel, and produce a completely random non-cyclic pattern (which is fine if this is the desired effect).

When explaining the drum machine to other people in conversation and mentioning the fact that it uses probabilities, they generally ask, "So it's completely random?" To which I inform them that no, it's not *completely* random, because using 100% values is basically the same as pushing down one of the step buttons on a conventional drum machine and having the sound play each time it loops around. The probability factor is mainly to allow for variety within a slightly repetitive loop, which, in essence, is basically what a human drummer does: Repeat the same loop over for one section (like the verse or head of a song), adding slight rhythmic, dynamic, and creative variations with each loop.

Subdivisions

After the 'main' function has decided whether it is going to play a sound or not (by calling the synth), it runs a second test to determine how many times it will play the sound within the 16th note. As with the step test, a value between 0 and 100 gets passed through to the main function to determine if the sound plays once (normally) or sub-divides the beat into 2 or more consecutive hits. For example, lets say the value 10 is passed through and the random test value generated is 7. 7 is less than 10, so a sub-division gets triggered. A sub-division value gets randomly chosen from a hard-wired weighted array, which at the moment is set to `[2, 3, 4, 5, 6, 12, 18, 24]`. If the value 4 is chosen, this means that the sound will be triggered 4 times in the space of one 16th note/step, which in musical terms is equivalent to a 64th note. Sub-divisions generally work better at slower tempos, and the higher values are not dissimilar to a human drummer's drum role.

MIDI Control

Most of the synth's arguments are controlled via a MIDI controller that sends out various MIDI control messages (usually with a range of 0 – 127, but not always), which then get scaled to the range suitable for each argument. Each instrument has the following dials: A buffer dial for changing the buffer and sound for the instrument, a pitch dial that changes the rate/pitch at which the sample plays, a decay dial that changes the envelope of the sound, and a level dial that determines the volume of the instrument. Each instrument also has two additional dials that control arguments passed to the 'main' function. The first one being a sub-division probability dial, and the other an effect probability dial. The effect probability dial is used to give a percentage of chance that a sound might be played with an effect on it, such as reverb, delay or something else.

Earlier versions of the sequencer didn't use a MIDI controller to control synth arguments and other values (mainly because I hadn't come to working on it at that stage), but rather, they used different arrays or random ranges to add dynamic and interesting variations to the patterns. For example, the decay on the kick or the other drums would be a random value, so some hits would be really short and others would be normal (or longer if the sound was mixed with reverb). The thing I missed the most from when the program worked in this way, was having a random rate value, ranged from say –2 to 2, basically meaning the snare would play at different pitches, forwards or backwards, with a weighted biased towards 1 (normal rate). It was nice to have the sound randomly jump around without having to twiddle a knob. I also used streamed arrays to add life to percussion patterns. For example, I had a cowbell that would loop with two short releases followed by a long release, which resulted in making it sound like it was being muted and then freed, in the same way a real percussionist might.

After playing around with one of the older uncontrollable versions, I thought to myself "why not just design it so it can be controlled or random? Both at the same time." After thinking about it for a while I thought the best way to do this would be to use the very last value on each dial as a switch to pre-configured random-choice arrays or randomly generated values. This works okay, the only problem being that if I want to tune the dial to the maximum control value, it's very hard to find the second last position easily.

Effects

Effects are created as other discreet synth definitions, which means there is one synth for clean sound, one for reverb, one for delay and so forth. 'Main' chooses between the synths by using a weighted array, which has variables that are changed by the effect probability dial. There are also two dials that work on instruments 2 (snare) and 5 (other) only. These dials control the backwards probability: The percentage of chance that a sample may play in the reverse direction. This works by changing the rate value to a negative number, so that a rate of –2 would play the sample backwards and an octave higher.

GUI

Step probability arrays can be changed while the sequencer is running by moving sliders on a GUI window. The window currently consists of two columns and five rows of multi-slider boxes that have either 16 or 32 sliders. Each instrument is on one row and the columns are for the 16 and 32 step arrays. The sliders load up in a default position (depending on the values in the step arrays) and can be adjusted reasonably quickly, drastically changing or just making slight adjustments to the patterns. I've also recently implemented a BBCut mode for generating complex break beat patterns.

BBCut is a SC3 Class Library created by Nick Collins. It basically takes a sound file, cuts it up and plays it back in a break beat granular-type style semi-random pattern. My program allocates a buffer that is exactly the length of one pattern cycle. Each time it plays through the cycle it records itself into the dedicated buffer. When it gets to the start of the next cycle it clears the buffer and starts recording again. When the BBCut button is pressed the program waits for the current cycle to finish (which means the buffer is also filled) and proceeds to loop around in BBCut mode – basically

playing really fast interesting beats using the audio recording from the drum machine. When the button is pressed again the program waits till the end of the loop and switches back to normal mode.

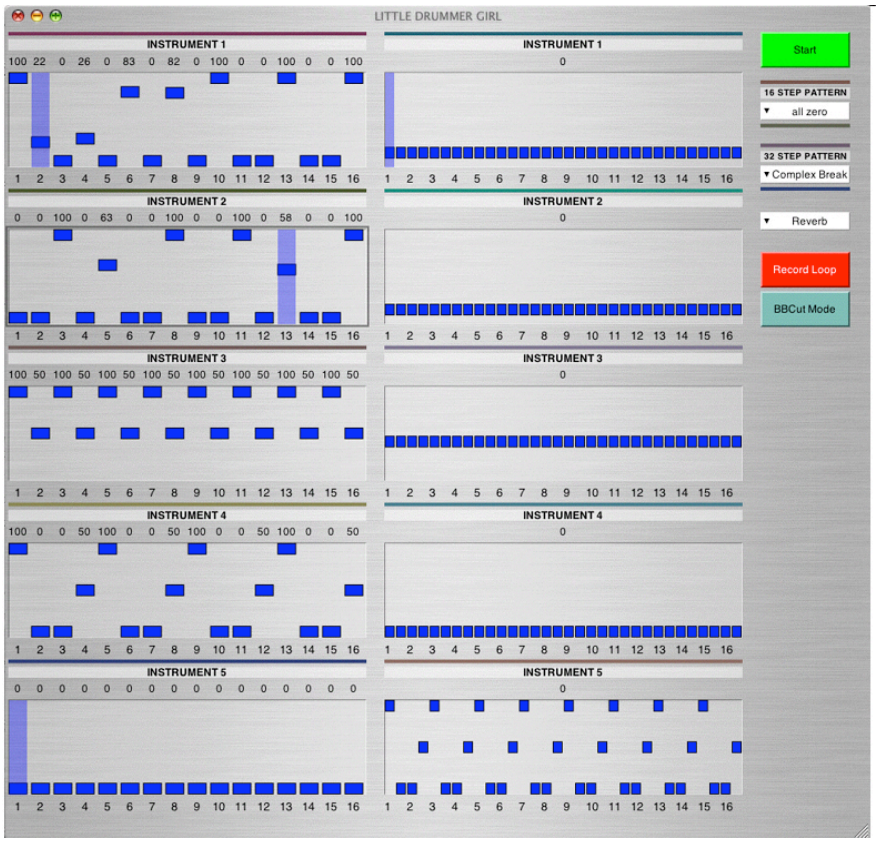

Figure 2. *An example of the GUI window.*

Integration with Live Instruments

Part of the project has involved 'jamming' with professional musicians in a fully improvised situation and testing out the ability of the machine to be played as an instrument while observing the amount of interaction that actually happens between myself and the other players. Each rehearsal has been digitally multi-tracked, meaning that I've been able to listen back to the performances without having to play and listen critically at the same time (Listening back to a recording of a performance is always different to listening while playing). The musicians I've been fortunate enough to play with (a bassist, James Luke and guitarist, Charlie Meadows) are both trained jazz players and have also had experience playing in drum and bass, funk, hip hop and dub styled groups.

The results of these 'jams' have been extremely rewarding and pleasing. Both musicians enjoyed it so much that they were keen to start rehearsing regularly and performing around town. The following are excerpts from questions I asked the musicians about the difference of playing with an interactive machine as opposed to a human drummer:

> I guess that besides the fact that [the] time doesn't move at all, you've got… it's almost like it's an alien sonic landscape. Normally there's fluctuation, and you do things, I don't know if it's consciously or sub-consciously, you slow down and speed up in sections, and it's not like that at all. So that kind of means you're playing almost less organically or something. I think it's really informing the music, obviously we're not playing like we would if it was a drummer and therefore the way that it climaxes and the way that we build tension or whatever is different as well… there's a different sort of interaction. (Charlie Meadows 2004)

> It's almost a slower interaction… kind of more subtle. (James Luke 2004)

This discussion brought up the point that although I've succeeded in developing a machine capable of using as an improvisational tool with an interactive edge (when compared to the conventional drum machine), the

type of interaction is quite different to that of a human drummer. I wouldn't say better or worse, just different.

Conclusion

Drum machines are here to stay whether we like it or not. New technologies will lead to the creation of better more powerful machines (Programs such as Stylus RMX are good examples of this). I believe future directions for this type of machine will involve fewer onuses upon the human as the decision maker, looking instead towards the power of the computer processor. A drum machine is envisioned capable of listening to and analysing its performance context; a machine capable of responding unaided yet musically to what other musicians play, in much the same way as a jazz drummer does, but without the hindrances of the human body and mind. This model presents a basic insight into the future of the drum machine in live popular and artistic music, looking towards the possibilities available after musical decisions are taken away from the human and given to the computer. These insights help display the necessity for movement into new rhythm creation techniques and hopefully will inspire more development in the area.

References

Garnett, G. 2001. "The Aesthetics of Interactive Computer Music" *Computer Music Journal*. 25, 1. 21-33.

Lewis, G. 2000. "Too Many Notes: Computers, Complexity and Culture in *Voyager" Leonardo Music Journal*. 10. 33-39.

McCartney, J. 2004. *The SuperCollider Home Page.* http://www.audiosynth.com/scfaq.html (April 2004).

McGuiness, A. 2004. *Microtiming Deviations in Groove*. MPhil thesis. ANU.

Moroni, B. Von Zuben, F. and Manzolli, Jônatas. 2002. "ArTbitration: Human-Machine Interaction in Artistic Domains" *Leonardo*. 35. 185-188.

Neill, B. 2002. "Pleasure Beats: Rhythm and the Aesthetics of Current Electronic Music" *Leonardo Music Journal*. 12. 3-6.

Valsamis, P. 2001. "Machines Drumming" *Convergence: The Journal of Research into New Media Art*. 7, 1. 61-73.