Return to Index

the final decision about what works as a groove or riff. The producer may finetune algorithmic composition methods to be consistent with their aesthetic needs for a particular track since the models can be adapted for specific circumstances.

Where decisions are taken, the expertise of the models are hard coded in the functions and data, so any choice from the available range for a parameter should be correct. We are not trying to search for a solution amongst competing constraints, but following rules precisely. There is no need for backtracking or any of the more complex apparatus of algorithmic composition.

There are no neural nets in this study, though it would be exciting to attempt to apply them say, in imitating successful breakbeats or breakbeat cuts.

2.1.1 Version one- probability templates

In this method we have a probability that a strike will be triggered for a particular time slot. A working set of probabilities (from 0.0 to 1.0) are:

```
//kick
[0.7, 0.0, 0.4, 0.1,    0.0, 0.0, 0.0, 0.2,    0.0, 0.0, 0.5, 0.0,    0.0, 0.0, 0.0, 0.3]
//snare
[0.0, 0.0, 0.0, 0.0,    1.0, 0.0, 0.0, 0.5,    0.0, 0.2, 0.0, 0.0,    1.0, 0.0, 0.0, 0.3]
//hihat
[0.0, 0.0, 1.0, 0.0    0.0, 0.0, 1.0, 0.7,    0.0, 0.0, 1.0, 0.0,    0.0, 0.0, 1.0, 0.7]
```

The reader will note that the snares on beats two and four always occur, whilst the kick does not have to sound at all in a given bar (we could of course use the generate and test paradigm and keep rolling dice until we get a run with at least one kick). The probabilities of each strike are all independent, though we could easily set up more complex transition tables where the probability of certain events changes based on what has already occurred. It is critical that each case, and especially the extreme cases are still valid beats. Snares on two and four with offbeat hats is the minimal beat, whilst the maximal beat with all events is not so busy as to be difficult to listen

A consequence of modelling dance music, in particular beats, is that we can attempt

ODDNUMBERS= {n integer: n is odd, and n<= SubDiv/2}

These give us our possible cut lengths. We can generate a random member of the set by this procedure:

```
Let TEMP= SubDiv/2 (integer division).
If TEMP is even, take MAX as (TEMP-2)/2, else MAX= (TEMP-1)/2
CHOICE = randominteger(0, MAX).
Convert CHOICE to an integer as CHOICE= 2*CHOICE +1.
```

Now that CHOICE has been created, we must make sure it fits the available number of units left in the phrase, Reduce CHOICE by two units a time until it fits into the remaining space.

Finally choose a number of repeats from 1 to MaxRepeats. If we overshoot the remaining space (by having too many repeats) then take a single block of size the remaining number of units. This gives the chance of a terminating segment, like the 2 in 3+3+2.

3.2 Results

Figure 1 demonstrates three partitions created from running the algorithm on three bars worth of units with SubDiv = 8. The filled blocks represent repeats, with a stutter effect finishing the first example.

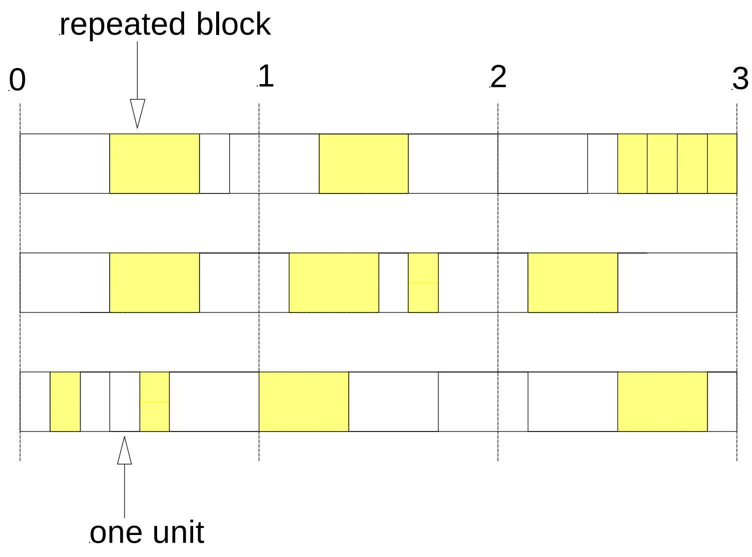

Figure 1 three bar, 24 unit partition examples

The model works really well on a SubDiv of 8. The syncopations are dynamic enough to be interesting, but not so complex as to lose the listener. The sense of time is not compromised by blocks that cut across bar lines, since the source sample is accessed modulo its own length. Since the model assumes the source sample is an integer number of bars long at the given tempo, then access beginning of the seventh eighth note of the first measure will sound relevant and correct. Because cuts in eighth notes are the stock in trade of style, the model has been optimised with this situation in mind.

3.4 Implementations

Implementations of the model have been made in SuperCollider for realtime use, and in a stand alone Win32 application for non-realtime (much faster than realtime) rendering. It would be relatively easy to set up implementations in Csound, or any other desired rendering language.

4 Conclusions

In this paper solutions were described to two tasks using algorithmic composition techniques, with application to, though by no means limited to, dance music. The construction of models gave analytical insight into dance music styles. Using music programming languages like SuperCollider allows the ready adaptation of this work to specific compositions, but specific processes could be packaged in a limited interface tool. The use of algorithmic composition systems for specific higher level composition tasks in home virtual studios will surely gain precedence in the next few years. The tools themselves will have their own distinctive features, making their use recognisable much like synthesis systems or sound processing tools can be distinguished. Hopefully, the tools will inspire new creative uses, and not just blind imitation of past masters.

5 Bibliography

5.1 Publications

[1] Ames, Charles. 1990. Statistics and compositional balance. *Perspectives of New Music*. 28:1.

[2] Shapiro, P. 1999. *Drum 'n' bass, the Rough Guide*. Rough Guides Ltd.

5.2 Web Sites

[3] software
Recycle- http://www.propellerheads.se
Wave Surgeon/Mobius- http://www.squarecircle.co.uk
SuperCollider- http://www.audiosynth.com/
Koan- http://www.sseyo.com/

[4] techniques
breakbeat cutting
good FAQ- http://www.dnbproduction.com/index.asp
very good techniques area- http://spinwarp.com/

general (some tips and tricks, discussions of groove)
http://www.dancetech.com/

6 Appendix- Pseudo C code for automatic breakbeat cutting

Local variables and variables persisting between calls to this function are not declared here.

```
//while loop to desired length output (non realtime),
//or scheduling itself for next decision(realtime)
{
if (unitsdone==totalunits)    //NEW PHRASE
{
//How many bars?
barsnow= randominteger(1, MaxPhraseLength);
//prepare units counter
totalunits= barsnow*SubDiv;
unitsdone=0;
repeatsdone=0; repeats=0;
}

unitsleft= totalunits- unitsdone;     //indirectly gives current time position

if(repeatsdone==repeats)   //NEW CUT
{
repeatsdone=0;
//How large a cut?

//ending a phrase with a repeating unit block? can only happen if less than a
bar left
if(unitsleft<SubDiv && (randomfloat(0.0, 1.0)< RepeatChance))
{
unitsincut=1;
repeats=unitsleft;
}
else
{
temp= SubDiv/2;

if(temp % 2==0) temp= (temp-2)/2;
else temp= (temp-1)/2;

unitsincut = randominteger(0, temp);

//turn unitsincut into an odd number less than SubDiv/2
unitsincut = 2* unitsincut +1;

//unitsleft is at least 1, and unitsincut is odd, so will terminate
while(unitsincut >unitsleft) unitsincut -=2;

repeats=randominteger(1, MaxRepeats);

//if overshoot on repeats, allow a terminating segment
if(repeats*unitsincut>unitsleft)
{
unitsincut= unitsleft;
repeats=1;
}
}

//prepare indexing info into the SourceSample.
//take current position within phrase modulo the SourceSample length.
//(for accurate breakbeat cutting, assumes that the SourceSample is
//in 4/4, and an exact number of bars long at the given tempo)
//store the sample indexing data so that each repeat can access the same
block
//CODE NOT REPRODUCED HERE
}

{	//CARRY OUT AN EVENT
//have source indexing data for this repeated block

//use unitsleft to get start time position, unitsincut gives duration

//RENDER BLOCK- CODE NOT REPRODUCED HERE

unitsdone += unitsincut;
repeatsdone += 1;
}

}
```