

「Real Time Risset Rhythm Generator」から「Live Sampling Risset Rhythm」への道 Towards the "Live Sampling Risset Rhythm" from the "Real Time Risset Rhythm Generator"

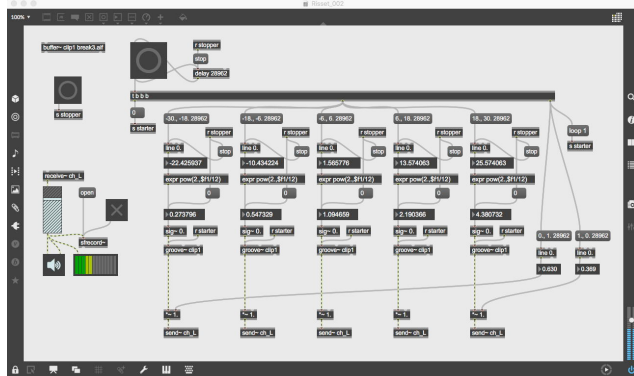
長嶋 洋一†
Yoichi Nagashima

1.はじめに

ピッチに関する有名な錯聴の"Shepard Tone"を、Jean-Claude Rissetがリズムに拡張した錯聴が"Risset Rhythm"である。筆者はMax8環境下での"Risset Rhythm"再構築として、リズムだけでなくピッチ要素を持つ音楽フレーズでの実装や、左右チャンネルで無限加速と無限減速する実験を行い[1]、さらに生成する"Risset Rhythm"のパターンをライブ変容させる「Real Time Risset Rhythm Generator」の提案を行なった[2]。そしてComputer Musicの領域で行われているライブ・サンプリングについて包括的に検討した[3]結果を受けて、本稿ではその先にある「常にテンポが連続的に加速/減速する"Risset Rhythm"音楽」のための手法へのアプローチについて検討報告する。

2.古典的Risset RhythmとサウンドファイルのRisset Rhythm化

Shepard ToneもRisset Rhythmもその基本的な原理は、多数のパーシャルの複合音としてサウンドを聴取する際に、各パーシャルの漸増/漸減がマスキングによって個別に注目(追従)できないという聴覚的な理由による[4]。ただしShepard Toneの場合にはそれぞれのパーシャルの長さや相互関係(ピッチ高低、時間的駆動の位相など)は比較的自由であり、いわばどのようにミックスしても簡単にShepard Toneが実現できた。

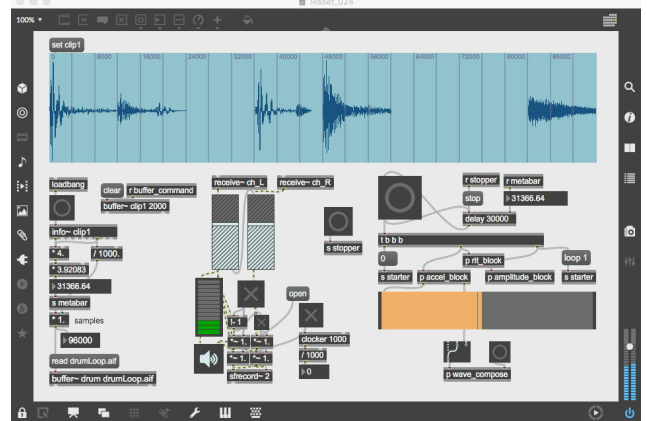


これに対してRisset Rhythmの場合には強い条件が求められる。例えば「無限加速」するループ音響を5つのバンド幅のパーシャルとして可変ピッチ再生させる場合には、上図のように3つの中間音域(-18半音→-6半音)・(-6半音→6半音)・(6半音→18半音)は全て指数関数的に上昇ピッチで再生し、さらに最低音域(-30半音→-18半音)ではフェイドイン、最高音域(18半音→30半音)ではフェイドアウトしつつ、指数関数的に上昇ピッチで再生すると、それぞれのループ音響の縁は隣接するループ音響と自然に連結する

ことで「無限加速」する(無限減速の場合には端点を交換して「下降」とする)。もっとも再生時間がかかる最低音域のサンプリングメモリの時間幅を"metaber"と呼び、最小単位ループの時間幅が小さい場合には4倍ないし2倍、6秒以上の長い音響サンプル(音楽フレーズ)の場合には1倍のままの時間幅に対して、およそ3.9218倍して経験的に30秒~50秒あたりの"metaber"とすると良好なRisset Rhythmサウンドが得られた。

3.Real Time Risset Rhythm Generator

単純なShepard Toneでなく「上昇しつつ下降するShepard Tone」という実験を受けて、単純なRisset Rhythmでなく「加速しつつ減速するRisset Rhythm」も追求したが、ステレオの左右で同時に無限加速と無限減速させた実験では、"metaber"のループ端点でステレオの中心に定位するという現象の他に面白い結果は得られなかった[4]。



そこで単調なRisset Rhythmを動的にリアルタイムアレンジする手法として、上のようにジャスト2msecの波形メモリを最小単位ループとしてその4倍"metaber"を基本的な枠組みと設定し、サウンド生成中の節目にランダム確率で、さらにマニュアルのタイミングで任意にも、それぞれリズム分割の妥当なタイミングを貼り込みポイント候補として、別のサウンドファイルから部分的に切り出したサウンド断片にリアルタイムで書き換えるという"Real Time Risset Rhythm Generator"を試作してみた[4]。

4.Computer MusicパフォーマンスとLive Sampling

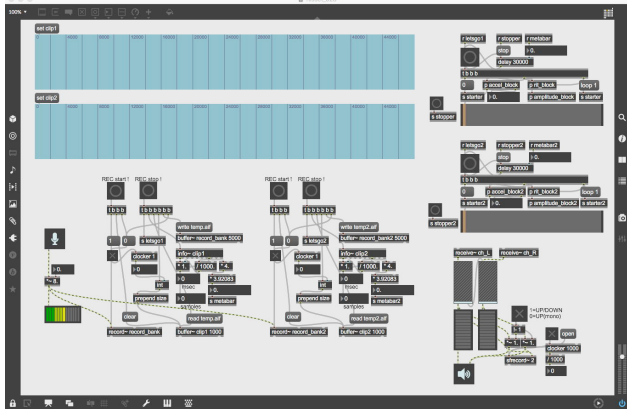
上の実験の結果、サウンドの変形は全て確率的にランダムな重み付けをしているので、書き換えたり書き換えなかったり、ある部分がしばし消えたり・・・を繰り返し、単調なRisset Rhythmよりは面白そうな可能性を確認できたものの、最小単位ループ2msecという枠組みは固定されているために、結局のところ「ちょっと面白い変形」程度であり、せっかく音楽フレーズのRisset Rhythm化にも成功していたのに、音楽そのものをライブに組み立てるといった基盤にはならなかった[4]。

そこで、これまでComputer Musicの領域で行われてきたライブ・サンプリングについて包括的に検討し[3]、その本質は自由なサンプリング長にあるという結論から、固定された波形メモリに基づくRisset Rhythm化のままでは駄目だという判断に至った。進めてきたRisset Rhythmシステムの「再生系」については、ライブ生成中に刻々と“metabar”パラメータを受けて変更しつつも動作を続けられるので基本的に問題がない事を確認したために、リアルタイム「録音系」の部分が焦点となってきた。

5. Live Sampling Risset Rhythmへの手法

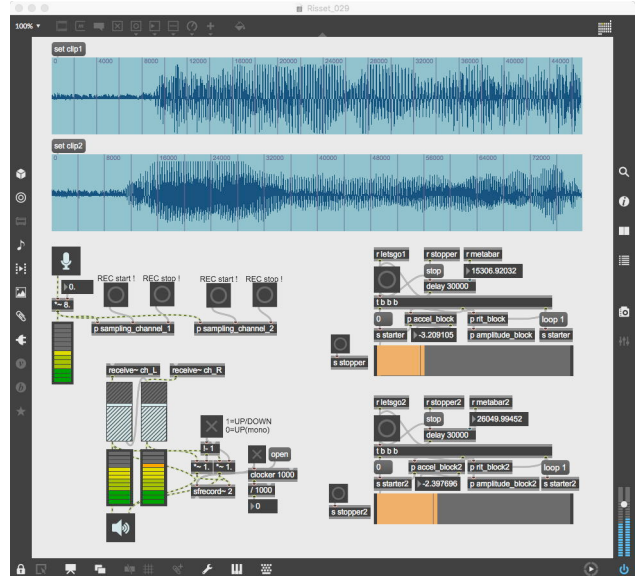
以下のMax8パッチはRisset Rhythm化のためのリアルタイム・サンプリングの実験中のMax8の様子であるが、重要な点として従来の「サンプリング(開始)」ボタンだけでなく、「サンプリング終了」ボタンを設けた点が画期的な進展である。これまでのComputer Musicシステムでは、サンプリングメモリはある程度の長さを持つ固定幅であり、センサ等による録音開始トリガでサンプリングすると、メモリを超える部分が単純に切り捨てられていたのに対して、このシステムでは録音終了のトリガが必須である。

これまで作曲の一部として多種のセンサやインターフェースや新楽器を制作して公演してきた[5]筆者にとって、このようなライブ・サンプリングのためのインターフェースというのは「お茶の子さいさい」である。例えばライブComputer Music作品“Berlin Power”(2002)[6]では、オリジナル開発したMIDI出力フットスイッチによってサンプリングを指示していたが、スイッチONだけでなくスイッチOFFも伝送可能なので、そのまま本システムに活用できる。



上のMax8パッチで実験的に実装したアルゴリズムはきわめてシンプルで、録音スタートのトリガに対して、まずclockerとループメモリを初期化する。続いてサンプリングメモリに録音を開始すると共にclockerで時間計測をスタートする。録音ストップのトリガに対しては、まずclockerを停止しつつサンプリングメモリへの録音を終了して、そのメモリ内容を「tempサウンドファイル」として書き出す。続いてclockerの数値(録音開始から終了までの経過時間)によってループメモリをリサイズしてから、そのループメモリに「tempサウンドファイル」を読み込む。これにより、「tempサウンドファイル」のサイズは十分に大きな5000msecとしているので、その冒頭の有効部分だけがちょうどループメモリに充填される。その後、ループメモリの情報からそのメモリサイズ(metabar)を取得してRisset Rhythm化の再生系に送って、再生(生成)をスター

トする。たったこれだけで、刻々と自在なサンプリング時間によって切り出した音響信号を生成単位とするRisset Rhythm化が実現できた。



上のMax8パッチは上述のサンプリング機構を盛り込みつつ、2系統のサウンドチャンネルで交互に「ライブサンプリング→Risset Rhythm化」を並行できるように試作した例である。試験的に実験してみたところ、「ライブ/自在に刻々と無限に加速し続ける音楽音響」を容易に生成することを確認できたが、そのような音楽というのはこれまで報告されておらず(音楽のテンポというのは大部分はほぼ一定、変化するとしてもたいていが有限の範囲内というのは常識)、この新しい概念のシステムはまさに新しい音楽の創出の可能性を秘めている。

6. おわりに

筆者の過去の活動[7]からすれば、こんなに面白いアイデアに到達すればまずは具体的な作品に適用して、ICMCやNIMEなど新しい音楽作品と音楽システムに関する国際会議にエントリーして発表/公演を目指すところである。ところが本稿執筆時点でまだ世界的にCOVID-19禍の只中であり、その機会は2022年まではお預けとなっているので、ここはじっくりと熟成させて作曲に取り掛かっていきたいと考えている。

参考文献

1. 長嶋洋一, Jean-Claude Risset 温故知新, 情報処理学会研究報告(2020-MUS-128), 情報処理学会(2020).
2. 長嶋洋一, Risset Rhythm 温故知新, 日本音楽知覚認知学会2020年秋季研究発表会資料, 日本音楽知覚認知学会(2020).
3. 長嶋洋一, メディアアートにおけるサウンド素材のライブ・サンプリング手法についての議論, 電子情報通信学会パターン認識・メディア理解研究会資料(技術研究報告)PRMU2021-01, 電子情報通信学会(2021).
4. <https://nagasm.org/ASL/RissetRhythm/>
5. <https://nagasm.org/ASL/>
6. https://www.youtube.com/watch?v=_5Emv7LXd9E
7. <https://nagasm.org/ASL/YouTube.html>