音楽情報科学 21－5
（1997．7．21）

“Improvisession”：ネットワークを利用した即興セッション演奏支援システム

長嶋洋一＊，中村文隆＊＊，片寄晴弘＊＊＊，井口征士＊＊＊＊

＊ LIST/神戸山手女子短大，＊＊ 神戸山手女子短大，＊＊＊ LIST/和歌山大，＊＊＊＊ 大阪大

ネットワークで相互接続された24台のUnixワークステーションを用いて、「アルゴリズム作曲」および「音楽における即興」について体験的に演習するための音楽教育システムを構築している。Ethernet上の24台のSGI Indy上にOpen-GLのGUI環境で対話的に即興演奏／音楽生成するプロセスを走らせて、相互の情報交換にRMCP(Remote Music Control Protocol)を用いた。本稿では、他の参加者に音楽情報をbroadcastしてセッションするモードに加えて、自分のマシンだけでモニタしながら簡易的なフレーズを生成するモードを加えた最新のバージョンについて、技術的な実現方法と音楽科の学生による演習の模様の報告を行う。

“Improvisession” : a performance support system for improvisational sessions with networks

Yoichi Nagashima*，Fumitaka Nakamura**，Haruhiro Katayose***，Seiji Inokuchi****

* LIST / Kobe Yamate College, ** Kobe Yamate college，*** LIST / Wakayama Univ.，**** Osaka Univ.

（ * email: nagasm@kobe-yamate.ac.jp ）

This paper describes a performance support system for improvisational sessions with networks. We call this system "Improvisession", and this paper reports its the latest version. The system runs with 24 SGI Indy computers through FDDI LAN, and softwares were developed with Open-GL, OSF/Motif, and RMCP (Remote Music Control Protocol) by C language. All performers can play their own phrases, and can broadcast their phrases to other performers with improvisation via RMCP. We report the technical points of this system and the experimental lectures of students of music course of Kobe Yamate College.

1. はじめに

神戸山手女子短期大学音楽科では1994年より「コンピュータと音楽」(長嶋) という講義において、24台のSGI社Indyワークステーションを用いたMultimedia作品制作演習・ホームページ制作演習およびComputer Musicについての紹介などを行っている[1]。

ここでは、Computer Musicの重要な要素であるAlgorithmic Compositionについて、あるいは「音楽における即興」についての議論を、公開されたMLによって学外の音楽家などと連携して進めている。また、センサを活用したMultimediaインタラクティブ・アート[2]を実際に体験することで、音楽演奏における「即興」について検討し、音楽活動における広がりの一つとしての理解・吸収を目指している。本年度は10月15日に神戸・ジーベックホールで行う音楽科公開講演会のコンサートの部において、実際に受講する学生がPerformerとして新作の上演に参加する予定である。

また1996年より、オリジナルの音楽教育システムとして、ネットワークを利用した分散処理による集団即興セッションの実現を目指した実験を開始した。"Improvisession" と名付けたこのシステムを実現するためのプラットフォームとしては、XのMotifを用いるとともに、映像系はOpen-GLによるCGを利用し、音響系は早稲田大学の後藤らが提唱[3]したRMCP(Remote Music Control Protocol)を採用している。

そして、これまでイメージ情報科学研究所で行ってきたIRIXのDigital Media Libraryを用いた開発環境[4]を活用しながら、RMCPパッケージのサンプルを利用し既存のMotif/X-GL環境のプログラムとして幾つかの試作プログラムを開発するとともに、受講する学生に実際に実験的に利用させて、体験的な感想などをMLで収集しながら仕様を改訂する、という実験的開発を進めてきた[5]。

2. 初期バージョンの動作と課題

第一段階として実験的に試作したプログラムでは、「X-Windows内のスライダーをマウスによって操作して、pitch、velocity、duration、panpot等を[演奏]する」というモードと、「画面内のボタンでランダム8音のフレーズを[演奏]し、画面内のスライダーで、音域、ランダム幅、インターバル等を指定する」という二つのモードを持つ簡易版であった。それぞれの「演奏」情報は各自のクライアントプロセスからbroadcastされ、あらかじめrootプロセスとして各マシンにバックグラウンドで走っているMIDI演奏サーバプロセスによって、それぞれがヘッドホンを経由してそれぞれのマシン毎のMIDI音源で聞くことができた。

この簡易バージョンでは、「カーソルキーでクロマティックに上昇させながらマウスボタンで逆転スキップ下降させる」などという、Motifならではの演奏ワザを開拓する学生も出現したが、全体としては多くの不満と注文が出た。もっとも多かった不満は、「多数が騒然とセッションすると、自分の音がどれだか判らなくなる」というものであった。また、パラメータを指定してランダムに生成する一種のフレーズであっても、たまたま気に入ったものについては自分のオリジナルとして記録して再利用したい、というリクエストもあった。シーケンサのような方向はもともと考慮していないシステムではあるが、簡易型のシーケンサの要求は多数あった。

3. 改訂新バージョンの仕様と動作

そこで本年度前期の「コンピュータと音楽」のために改訂バージョンにおいては、以下のような要素を盛り込んで毎週少しずつ改訂した版を学生に提供し、実験とともに感想やさらなるリクエストを求めた。

3-1 マウスによる演奏とbroadcast

以前の簡易型の「スライダーをマウスで左右に動かす」という演奏があまり好評ではなかったので、画面内に具体的にマウスのアイコンをCGによって表示し、これをマウスの操作と完全に一致させて動かし、マウスの左ボタンによって音を出す、というモードに変更した。マウス指定範囲において、左ボタンを押す位置の横方向（x座標）が発生する音のピッチに、そして縦方向（y座標）が発生する音の音量に反映される。

それぞれの音は画面右側の8種類の音色から選択できるとともに、画面下の8個のパラメータ設定用スライダーのうち、Duration設定用スライダーの設定に応じたON-OFF時間の音として発音される。

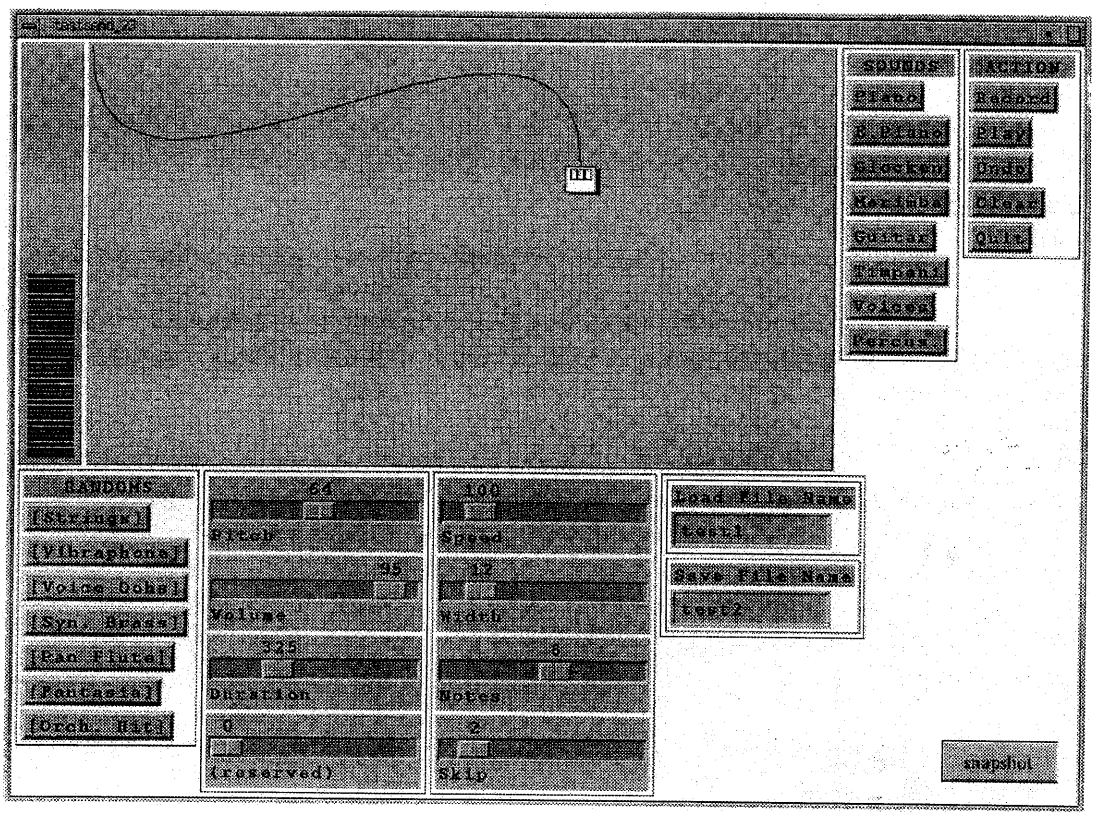
これは、筆者の一人（長嶋）がかつてマルチメディア・インタラクティブ・アート作品 "Muromachi" シリーズ [6] のエミュレータとして Open-GL によって開発していたものを組み合わせただけであり、Motif の開発環境の親和性によって簡単に実現できた。

3-2 ランダム生成フレーズの演奏と broadcast

この部分は、簡易型のバージョンとほとんど変化がない。設定できるのは、

i. スタートボタンによる 7 種類の音色の選択
ii. [Pitch] スライダーによる、フレーズの基準音域
iii. [Volume] スライダーによる、フレーズの音量
iv. [Duration] スライダーによる、個々の音の ON-OFF 時間の長さ
v. [Speed] スライダーによる、個々の音の生成間隔時間
vi. [Width] スライダーによる、ランダムノート生成のレンジ
vii. [Notes] スライダーによる、フレーズ内の音の個数
viii. [Skip] スライダーによる、ランダム生成の音の間隔（半音の＊倍）

というパラメータであり、詳細に設定すると、かなり表現の異なる印象のランダムフレーズを生成できた。



3-3 オリジナルフレーズの実験モード

簡易版でもっともリクエストの多かったのは、「全てのアクションが全員に聴かれてしまうのでなく、自分だけで実験して、出来上がったものを他人に伝えたい」というものであった。

そこで、マウスの右ボタンについてはRMCPによってbroadcastするのでなく、直接にIndyのMIDIポートを叩いて自分の音源に演奏情報を伝えることで、いわばリハーサルを行えるようにした。音色とDurationは画面のパラメータに対応するが、ボタンを押したままでのドラッグに対応した連続演奏についてはRMCPを経由しないため、未対応とした。

3-4 オリジナルフレーズの記録モード

オリジナルフレーズの実験モードを実現して提供したところ、多数の学生から「せっかく作ったフレーズを記録しておきたい」というリクエストが殺到した。これはもっともな要請であり、ここで簡単な一種のフレーズシーケンサ機能を盛り込むこととなった。

マウス画面の左にシーケンスデータのデータ数を示すバーグラフを設けて、最大でおよそ80音のフレーズまで記録できる。操作ボタンは

i. [Record] ボタン：ここから記録開始（以前のデータをクリア）
ii. [Play] ボタン：記録したフレーズをbroadcastしつつ再生する
iii. [Undo] ボタン：直前の1音の記録をキャンセルしてポインタを戻す
iv. [Clear] ボタン：記録データをクリア（ポインタをスタートに）

だけである。[Record] ボタンを押すとバーグラフ背景色が記録モードに変化し、ここではマウスの中央ボタンによって1音単位で記録できる。この簡易シーケンサに記録される情報は、

i. その音のピッチ（x座標）
ii. その音の音量（y座標）
iii. その音の音色
iv. その音のDuration

の4種類だけであり、Durationは2バイト量なので内部的には1音に5バイトを配列として割り当てているだけの簡単なものである。

なお、個々の音ごとに異なる音色や異なるDurationを割り当てて一つのフレーズを構成する、というのは、「音群技法による作曲手法」[7]の実現例となっている。また、フレーズに必要な「休符」については、極端に音量を下げた音符、ということで代用しているが、実用上はまったく問題ない。

3-5 フレーズのファイル登録と呼び出し演奏

せっかく作ったフレーズがシステムの終了とともに消えるのは悲しい、という悲痛なリクエストから、簡易シーケンスデータに任意のファイル名をつけてセーブし、また任意の名称のファイルから本システムのデータと認識して読み込むモードを追加した。このため、ファイルの先頭には認識用のヘッダ情報を付加した。

読み込み指定によってフレーズ記録ファイルが正しく読み込まれた場合には、データ数を示す画面に表示が現われて、そのままRocordモードとして音を追加したり、Playボタンでこのフレーズをbroadcast再生できるようになっている。

4. システムの実現方法と技術的検討

4-1 マルチメディア開発環境

システム全体の記述はIRIXの標準的なDigital Media Libraryを用いた開発環境を利用して、C言語で行っている。GUI部分はMotif Toolkitを活用し、CGの部分は通常はOpen-GLで記述するが、ここでは特にX-GL

ライブラリを利用して、完全にCG部分を一つの構成部品として取り込み、後述するMIDI環境と統合したマルチメディアシステムとしている[8]。

4-2 RMCPを利用したマウスとランダムの演奏

マウスの左ボタン、およびランダム生成によるフレーズ演奏には、RMCPパッケージに付属していたサンプルプログラムを活用している。ここでは、

```
void random_play( int ch )
{
        int i, wide, scop;
        u_char buff[16];
        long ctime, stime;
        stime = get_RMCP_ctime();
        for(i=0; i<notes; i++){
                ctime = stime + (long)i * (long)speed;
                CONV_RMCP_MSEC2TSTAMP(ctime, RMCP_cts.sec, RMCP_cts.msec) ;
                buff[0] = 0x90 + ch ;
                if( (range+width)<104 ) wide = width;
                else wide = 104 - range;
                scop = (int)random() % wide;
                buff[1] = skip * (scop / skip) + range;
                buff[2] = velocity;
                send_RMCP_smidi_buf(buff, 3) ;
                ctime += (long)duration;
                CONV_RMCP_MSEC2TSTAMP(ctime, RMCP_cts.sec, RMCP_cts.msec) ;
                buff[2] = 0;
                send_RMCP_smidi_buf(buff, 3) ;
        }
}
```

というような簡単な記述で、各パラメータに応じたランダムフレーズが生成できる。これらのパラメータはマウスを監視するルーチンを記述することなく、各ボタンやスライダー等のウィジェットを定義した初期設定部分のコールバックルーチン内で設定／変更される。

4-3 RMCPとMIDI処理ルーチンの混在

今回の改訂においてもっとも懸念されたのは、同じマシンのMIDIポートを利用するRMCPサーバ（RMCPss）と、プログラム自身が直接MIDIポートを叩く場合との競合であった。しかしこれは実験してみると、何の問題もなく共存できることが確認できた。そこで、broadcastする時にはRMCPを利用し、個別にプライベートに実験する際には直接MIDI制御、という使い分けを行った。

なお、筆者の環境およびRMCPの対応しているIRIXバージョンは5.3であるが、ここにはMIDI処理のシステム部分にバグがあり、マニュアル通りのプログラムではMIDI処理が正常に行われない。そこで筆者はこのライブラリのバグを解析して、以下のようにパッチを当てたオリジナルMIDI処理ルーチンを愛用している。

```
#define MES(x)   ( (x[0]).mm.msgbuf )
#define midi_set_status(x,d)    ( MES(x) = ( ( d > 0xbf ) && ( d < 0xe0 ) ) ? 0x40000000 : 0x60000000, \
                                MES(x) &= 0xff0fffff, MES(x) |= ( ( d & 0xf0 ) << 16 ) )
#define midi_set_channel(x,d)  ( MES(x) &= 0xfff0ffff, MES(x) |= ( ( d & 0x0f ) << 16 ) )
#define midi_set_keyno(x,d)    ( MES(x) &= 0xffff00ff, MES(x) |= ( ( d & 0x7f ) << 8 ) )
#define midi_set_velocity(x,d) ( MES(x) &= 0xffffff00, MES(x) |= ( d & 0x7f ) )
MIport *midi_port;   MIevent midi[1];

void midi_initialize(){
        MIconfig *c;   u_int pbuf[8];
        c = MInewconfig(); midi_port = MInewport();
        if( MIopen( midi_port, "w", &c ) < 0 ) exit(-1);
}

void midi_transmit( int status, int channel, int keyno, int velocity ){
        midi_set_status( midi, ( status & 0xf0 ) ); midi_set_channel( midi, ( channel & 0x0f ) );
        midi_set_keyno( midi, ( keyno & 0x7f ) ); midi_set_velocity ( midi, ( velocity & 0x7f ) );
        if( MIsend( midi_port, midi, 1 ) < 0 ) exit(-1);
}
```

```
void midi_tx_3byte( int status_byte, int keyno, int velocity ){
        midi_transmit( ( status_byte & 0xf0 ), ( status_byte & 0x0f ), keyno, velocity );
}

void direct_midi( int status, int keyno, int vel ){
        (midi[0]).dt.opaque1 = 0L; (midi[0]).dt.opaque2 = 0L;
        midi_tx_3byte( status, keyno, vel );
        (midi[0]).dt.opaque1 = (long)duration / 1000;
        (midi[0]).dt.opaque2 = ((long)duration % 1000) * 1000;
        midi_tx_3byte( status, keyno, 0 );
}
```

5. 実験の状況と学生の反応

このように頻繁に改訂されるバージョンのシステムで実験を行う、という過酷な環境であるが、学生の大部分はリクエストに対応して成長するソフトウェア、というあまり他で体験できない環境を楽しんでいるようである。もともとクラシック音楽一辺倒の音楽科の中でまったく浮いている特殊な講義であり、講義登録の際に「柔軟な感性と視点を持つ学生のみ募集」と告知していることもあって、あまり大きな抵抗感もなく実験が進んでいる。

6. むすび

本稿では、ネットワークで相互接続された「アルゴリズム作曲」および「音楽における即興」について体験的に演習するための音楽教育システム構築の進展状況について報告した。

この実験はまだ始まったばかりであり、今後も講義の進展とともに成長させていくシステムであると考えている。また、ソフトウェアだけでなく、ヒューマンインターフェースとしてのオリジナルセンサについても積極的に開発していき、機会あるごとに学生の参加する具体的な作品として実現していきたいと考えている。

なお、この実験の状況と音楽的な検討、さらにツール自体については、筆者の一人（長嶋）のホームページ（http://www.kobe-yamate.ac.jp/personel/nagasm/）やMLを通じて広く公開していく予定である。今後も興味ある人々との情報交換を進めて、新しい音楽教育の可能性を追求していきたい。

謝辞

本研究の一部は、(財) 中山隼雄科学技術文化財団の研究助成を受けて行われた。ここに謝意を表する。

参考文献

[1] 長嶋洋一, 中村文隆, 稲松千奈美, 渡辺卓也：マルチメディア・ワークステーションによる情報基礎教育の試み. 情報処理学会平成6年度後期全国大会講演論文集I, pp.5–6, 1994.

[2] Y.Nagashima, H.Katayose, S.Inokuchi : A Compositional Environment with Interaction and Intersection between Musical Model and Graphical Model — "Listen to the Graphics, Watch the Music" —. Proceedings of 1995 International Computer Music Conference, pp.369–370, 1995.

[3] 後藤真孝, 橋本祐司：MIDI制御のための分散協調システム—遠隔地間の合奏を目指して— 情報処理学会研究報告 Vol.93,No.109 (93-MUS-4), pp.1–8, 1993.

[4] 長嶋洋一, 片寄晴弘, 由良泰人, 藤田泰成, 井口征士：マルチメディア生成系におけるプロセス間情報交換モデルの検討. 情報処理学会研究報告 Vol.95,No.74 (95-MUS-11), pp.63–70, 1995.

[5] 長嶋洋一, 中村文隆, 後藤真孝, 片寄晴弘, 井口征士：ネットワーク上で相互作用するアルゴリズム作曲系を用いた音楽教育システム. 情報処理学会平成9年度前期全国大会講演論文集II, pp.273–274, 1997.

[6] 長嶋洋一：Multimediaパフォーマンス作品 "Muromachi". 京都芸術短期大学紀要〔瓜生〕第17号1994年, pp.39–43, 1995.

[7] 長嶋洋一：音群技法による音楽作品のための演奏支援システム. 情報処理学会平成2年度後期全国大会講演論文集I, pp.253–254, 1990.

[8] Y Nagashima : Multimedia Interactive Art : System Design and Artistic Concept of Real-Time Performance with Computer Graphics and Computer Music. Proceedings of Sixth International Conference on Human-Computer Interaction, pp.89–94, 1995.