卒業研究

平成21年 3月5日

コード進行を考慮したweb上における自動作曲システム

5年　情報工学科　9番
木口　寛之
指導教員　東海林　智也

目次

1. 目的

近年，コンピュータや作曲ソフトウェアが簡単に入手出来るようになったことで，楽器が演奏出来ない人でも趣味で簡単に作曲が出来るようになった．しかし，人が聴いて違和感の無い自然な曲を作るには，多くの難解な音楽的知識を身につけなければならず[1][2]，学習に時間がかかってしまうため挫折してしまう人も少なくない．そこで，本研究では音楽的知識の有無に関わらず，誰でも簡単に作曲が出来るような自動作曲システムの開発を行う．

従来の自動作曲を行う手法には様々な種類がある．例えば，既存の曲からリズムやコード進行，音符の千位などを取り出し知識ベースを作成し，それをもとに作曲を行う「知識ベースを用いる手法」[3]，あらかじめ音符の遷移やリズムの変化のパターンを決めてデータベースを作成し，そのパターンをランダムあるいはルールに基づいて組み合わせて作曲する「用意されたパターンを組み合わせる手法」[3]，乱数等の数値データを音楽理論として用意したルールを用いて音符やリズムなどに換えて作曲する「数値データを与えて音楽理論に基づいて編集する手法」[3]などである．最近では多項式などの数学的手法を利用した自動作曲の研究[3]や，脳波はDNAなどの生体のゆらぎと呼ばれるものを用いた自動作曲の研究や[4]，遺伝的アルゴリズムを用いた自動作曲も行われている[5]．多くの場合，これらの手法を単体で使用するのではなく，複数の手法を組み合わせて自動作曲を行う．

我々がこれまで行ってきた従来研究では，「知識ベースを用いる方法」を自動作曲の方法として選択した[6][7]．その理由としては，知識ベースを作成する際に使用した曲の特徴を保存することが出来るため，ある音楽ジャンルに関する知識がない人でも，曲を作成する際に音楽ジャンルを考慮する必要が無いからである．しかし，我々が行ってきた従来研究ではコード理論[8][9]等の作曲の際に考慮しなければならないルールが無視されていたため，作成されたメロディは音階の遷移が不自然なものであり，自然な曲とは言えないものであった．

そこで本研究では，「知識ベースを用いる方法」を自動作曲の手法として選択しつつ，従来研究で考慮されていなかったコード理論を新たに取り入れた自然なコード進行の作成を目指す．ユーザの嗜好を反映したメロディを作成するため，コード進行パターンをPC上で再生しユーザの嗜好情報を収集する．収集した嗜好情報からコード進行の度数分布表を作成し，MIDI形式のファイルを出力する．なお嗜好情報の収集時間短縮ためwebブラウザを用いてユーザの嗜好を収集する．

本論文の構成

1章では本研究の目的や従来の自動作曲の手法を説明する.

2章では本研究で作成する自動作曲システムの概要や，ユーザの嗜好情報の収集に使用するwebアプリケーション，および出力されるコード進行のファイル形式であるMIDIについて説明する.

3章では本研究で考慮するコード理論や，使用するコード進行パターンについて説明する.

4章では最終的に出力されるコード進行の作成方法や，ユーザによる評価コード進行の度数分布について説明する.

5章では本研究で得られた結果について考察する.

6章では本研究での改善点について説明する.

Automatic Composition System by Reinforcement Learning of Chord Progression

Nowadays, a lot of software for musical composition is on sale, and even a person who isn't able to play musical instruments can write ~~a~~ music by using PC. However, ~~a~~ broad musical knowledge is necessary to compose a natural melody. For that reason, we develop an automatic composition system by reinforcement learning of chord progression. The proposed system enables everyone to compose a music easily.

Key words: composition, chord progression

2. 自動作曲システムの構築

自動作曲システムを構築するにあたって，本研究では「知識ベースを用いる方法」を自動作曲の手法として利用する．昨年度までの従来研究で使用されていた強化学習を使わず，代わりにコード理論を取り入れる．

2.1 システム概要

本研究で作成する自動作曲システムの流れを以下に示す．本システムはPHPを用いて作成した．OSはCentOS 4.0，Webサーバは Apache 2.0.52，PHPのバージョンは4.3.9を使用した．

1. ユーザネームを入力する（図1）．

2. webブラウザ上でコード進行パターンの音声が出力される．

3. ユーザは出力されたコード進行パターンを聞いて，「好き」か「嫌い」かを選択する（図2）．

4. システムはユーザの「好き」か「嫌い」かを0と1に数値化する．

5. 全コード進行パターン(62パターン)の出力が完了していない場合2.へ，完了した場合は6.へ進む．

6. システムはユーザが入力した各コード進行パターンの評価を全てまとめたCSVファイルを出力する．

7. システムはC言語で作成したプログラムを実行し(付録1)，6.で出力されたcsvファイルからユーザ好みのコード進行をMIDI形式で作成する．

8. webブラウザ上で7.で作成されたMIDIファイルが演奏される（図3）．

図 1.　ユーザネーム登録画面

図 2.　コード進行パターン評価画面

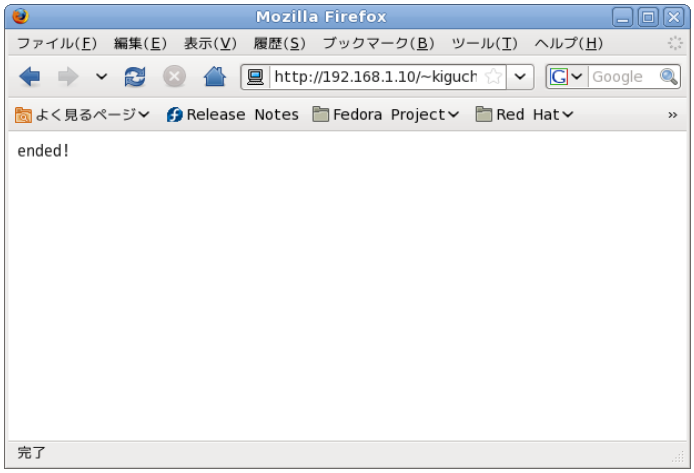

図 3.　終了画面(同時に MIDI の演奏)

2.2 PHP について

PHP（PHP: Hypertext Preprocessor）はスクリプト言語であり，動的に web ページを生成する web サーバの拡張機能の一つである[10][11][12]．本研究ではこの PHP を用いて，ユーザからのコード進行パターンについての嗜好情報の入力フォームを作成した(付録 2～4)．

2.3 MIDI について

MIDI（Musical Instrument Digital Interface）[13][14]とは，シンセサイザや電子ピアノ等の電子楽器やコンピュータ間での演奏データや音色データを送受信するための世界共通のインターフェース規格である．本研究で作成する自動作曲システムでは，コード進行を MIDI 形式のファイルとして出力する．

3.　コード理論について

昨年度までの先行研究のアルゴリズムでは，コード理論[15][16][17][18]等の作曲の際に考慮しなければならないルールが無視されていたため，作成されたメロディの遷移が不自然なものであった．そのため本研究では新たにコード理論[19][20]を取り入れる．

3.1　コードとは

コード(和音)とは，高さの異なる2つ以上の音の組み合わせである．例えばCメジャーコードなら「ド」，「ミ」，「ソ」の3つの音から構成される．

本研究ではCメジャーのキー(調)で主に使用されるコード7種類(表1)からコード進行を作成する．トニックには安定感があり，サブドミナントには不安定な感じを受ける．ドミナントには緊張感があり，トニックへ移ろうとする傾向がある．

トニック	ドミナント	サブドミナント
C	G7	Dm7
Em7	Bm7	F
Am7		

表1.　本研究で使用するコード一覧

3.2　コード進行パターンについて

コード進行には以下の様な基本的な法則がある[21][22][23][24].

1. 必ずトニックで始まる.
2. トニックからドミナント，サブドミナントへ進行可能.
3. サブドミナントからトニック，サブドミナントへ進行可能.
4. ドミナントからサブドミナントへは進行不可.
5. 必ずトニックで終了する.

これらを図にすると次の図4で表される.

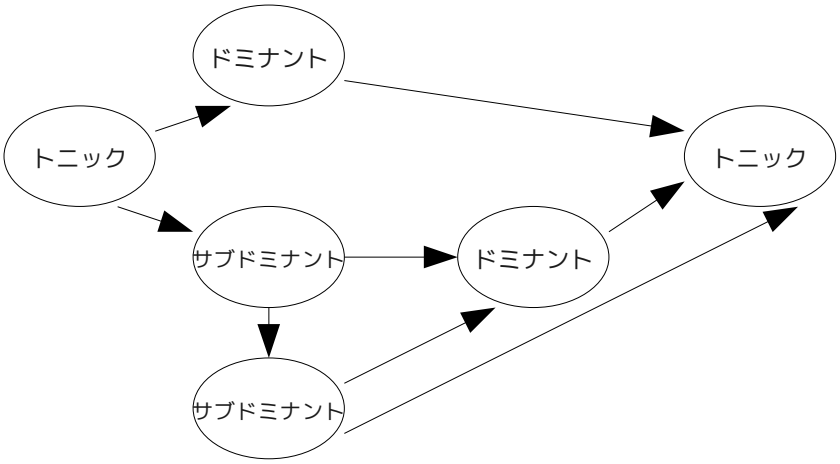

図.4　コード進行の状態遷移図

本研究ではこれらの法則に基づき，表1のコードの中から3つのコード進行の組み合わせを62種類考えた(表2〜6).

コード1	コード2	コード3
トニック	ドミナント	トニック
C	G7	C
C	G7	Em7
C	G7	Am7
C	Bm7	C
C	Bm7	Em7
C	Bm7	Am7
Em7	G7	C
Em7	G7	Em7
Em7	G7	Am7
Em7	Bm7	C
Em7	Bm7	Em7
Em7	Bm7	Am7
Am7	G7	C
Am7	G7	Em7
Am7	G7	Am7
Am7	Bm7	C
Am7	Bm7	Em7
Am7	Bm7	Am7

表2.　使用するコード一覧（トニック-ドミナント-トニック）

コード１	コード2	コード３
トニック	サブドミナント	ドミナント
C	F	G7
C	F	Bm7
C	Dm7	G7
C	Dm7	Bm7
Em7	F	G7
Em7	F	Bm7
Em7	Dm7	G7
Em7	Dm7	Bm7
Am7	F	G7
Am7	F	Bm7
Am7	Dm7	G7
Am7	Dm7	Bm7

表3.　使用するコード一覧（トニック-サブドミナント-ドミナント）

コード１	コード2	コード３
サブドミナント	ドミナント	トニック
F	G7	C
F	G7	Em7
F	G7	Am7
F	Bm7	C
F	Bm7	Em7
F	Bm7	Am7
Dm7	G7	C
Dm7	G7	Em7
Dm7	G7	Am7
Dm7	Bm7	C
Dm7	Bm7	Em7
Dm7	Bm7	Am7

表4.　使用するコード一覧（サブドミナント-ドミナント-トニック）

コード１	コード2	コード３
サブドミナント	サブドミナント	トニック
F	F	C
F	F	Em7
F	F	Am7
F	Dm7	C
F	Dm7	Em7
F	Dm7	Am7
Dm7	F	C
Dm7	F	Em7
Dm7	F	Am7
Dm7	Dm7	C
Dm7	Dm7	Em7
Dm7	Dm7	Am7

表5.　使用するコード一覧（サブドミナント-サブドミナント-トニック）

コード１	コード2	コード３
サブドミナント	サブドミナント	ドミナント
F	F	G7
F	F	Bm7
F	Dm7	G7
F	Dm7	Bm7
Dm7	F	G7
Dm7	F	Bm7
Dm7	Dm7	G7
Dm7	Dm7	Bm7

表6.　使用するコード一覧（サブドミナント-サブドミナント-ドミナント）

3.3　サンプリングについて

自動作曲システムで使うコード進行パターンの音声ファイルはICレコーダー(TASCAM DR-1)（図5）を用いてエレキギター(STEINBERGER製)の音をwav形式で録音した．その後，音量をノーマライズしてswf形式のファイルに変換した．なお，コード進行パターンの録音時間は，ユーザの嗜好情報入力の作業時間の短縮のため，1つあたり3秒程度とした．

図5.　本研究で使用したICレコーダー（TASCAM DR-1）

4.　コード進行の作成

昨年度までの従来研究では，既存の曲から音階の遷移回数を記録し度数分布を作成していたが，本研究ではユーザのコード進行パターンの好みから度数分布を作成する．

4.1　ユーザによる評価

前述した嗜好情報入力フォームより，ユーザは各コード進行パターンに対する評価を入力する．ユーザが再生されたコード進行パターンを「好き」と評価した場合は 1，「嫌い」と評価した場合は 0 がユーザの評価に与えられる(表 7)．

コード 1	コード2	コード 3	ユーザの評価
C	Dm7	C	**1**
C	F	Dm7	0
Dm7	C	Dm7	**1**
Dm7	F	C	**1**

表 7.　ユーザによって評価されたコード進行の評価の例

4.2　コード進行の度数分布

ユーザが入力したコード進行パターンの評価を CSV ファイルに出力し，度数分布を作成する．CSV ファイルのユーザの評価が 0 の場合は無視し，1 の場合は度数分布を更新する．例えば表 7 の 1 列目なら，度数分布の C 行 Dm7 列，Dm7 行 C 列にそれぞれ 1 が加算される(図 6)．ここで図 6 の度数分布の行は現在のコード，列は次に進むコードを表す．

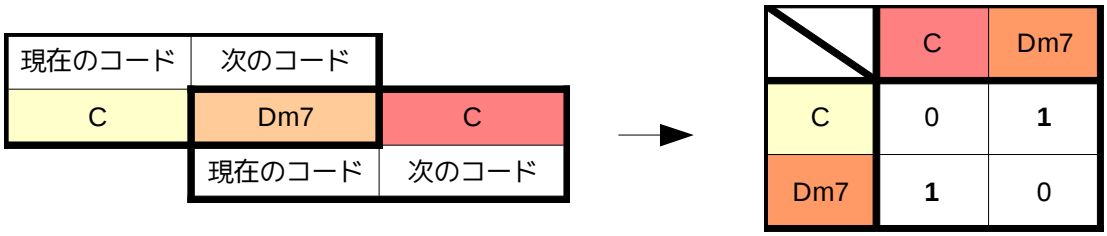

図 6.　度数分布の作成

表 7 から度数分布を作成すると最終的に表 8 のようになる.

	C	Dm7	Em7	F	G7	Am7	Bm7
C	0	2	0	0	0	0	0
Dm7	2	0	0	1	0	0	0
Em7	0	0	0	0	0	0	0
F	1	0	0	0	0	0	0
G7	0	0	0	0	0	0	0
Am7	0	0	0	0	0	0	0
Bm7	0	0	0	0	0	0	0

表 8.　表 7 から作成した度数分布表

5. 実験

5.1 実験概要

同研究室員5人から各コード進行パターンに対する嗜好情報を収集しCSVファイルを作成した．また，CSVファイルから作成した度数分布表からユーザの嗜好を反映したMIDIファイルを出力した．

5.2 結果

本研究で出力されたコード進行はこれまで我々が行ってきた自動作曲の研究に比べより自然なものとなったが，ユーザ評価するコード進行パターンのバリエーションが少ないため，出力されるコード進行にはあまり大きな差が見られなかった．

5.3 考察

現在考慮しているコード進行パターンはCメジャーだけなので，あらゆるユーザの好みに対応しきれているとは言い難い．更に三つのコードで一つのコード進行パターンを構成しているため，ユーザがそのコード進行パターンが好きか嫌いか判断する事が困難であった．解決策として，始めにユーザにキーの選択をさせる事などが考えられる．

6. まとめ

本研究で作成されたコード進行は，これまで我々が行ってきた自動作曲の研究と比べ考慮していなかったコード理論を取り入れたため，より自然なものとなった．しかし，提案手法でコード進行を作成する上でコード進行のバリエーションが少ないという問題点が出てきた．

しかしコード進行のパターンやキーを増やすと，ユーザがコード進行パターンの好みを評価し終えるのに時間がかかってしまう問題が生じる．一つのコード進行パターンあたりの評価時間が 8 秒程度あり，かつパターンが 62 種類あるため，ユーザの嗜好情報の収集時間が平均して 7 分から 8 分程度かかってしまった．

以上のどちらか一方の問題を解決しようとすると，もう一方の問題が大きくなってしまう．そこで今後の課題としては，始めにユーザにキーの選択をさせたり，不必要なコード進行をカットするなどして，コード進行パターンのバリエーションの増加と嗜好情報収集時間の短縮の両立を目指す．

参考文献

[1] デイヴ・スチュワート：絶対わかる！曲作りのための音楽理論,(リットーミュージック， 2006年).
[2] 福田 裕彦：誰も書かなかった作曲の「タネあかし」,(ヤマハミュージックメディア， 2007年).
[3] 勝田 哲司：月刊DTMマガジン10月号 シリーズ 自動作曲研究所 乱数で音楽を作ろう！,p.57,(寺島情報企画， 1999年).
[4] 勝田 哲司：月刊DTMマガジン1月号 シリーズ 自動作曲研究所 1/fのゆらぎでメロディを作ろう！,p.112,(寺島情報企画， 2000年).
[5] 今井 繁,長尾 智晴：遺伝的アルゴリズムを用いた自動作曲,(東京工業大学 工学部 像情報工学研究施設， 1998年).
[6] 熊谷 洋希：Q学習を用いたメロディの作成,(平成18年度　情報工学科卒業研究).
[7] 小島 京輔：Profit Sharingを用いた自動作曲,(平成19年度　情報工学科卒業研究).
[8] 若松 正司：若松正司の音楽セミナー　コードの使い方1,(音楽之友社， 1989年).
[9] 成瀬 正樹：コード進行スタイル・ブック,(リットーミュージック， 2001年).
[10] KJ,田中 ナルミ：PHPによるWebアプリケーションスーパーサンプル,(ソフトバンク クリエイティブ株式会社， 2006年).
[11] 山田 祥寛：PHP事典,(秀和システム， 2006年).
[12] 鹿又 広行：音楽的特徴と嗜好情報に基づいた楽曲推薦システムの構築,(平成19年度　情報工学科卒業研究).
[13] 高橋 信之：コンプリートMIDI　プログラミング・ブック,(リットーミュージック， 2006年).
[14] 御池 鮎樹：裏口からのMIDI入門,(工学社， 2002年).
[15] 北川 祐：絶対わかる！コード理論,(リットーミュージック， 1997年).
[16] 北川 祐：絶対わかる！コード理論2 ダイアトニック・コードのすべて,(リットーミュージック， 2006年).
[17]北川 祐：絶対わかる！コード理論3　ドミナント・コードのすべて,(リットーミュージック， 2006年).
[18] 篠田 元一：実践コード・ワークComplete　アレンジ編,(リットーミュージック， 2005年).
[19] 水谷 友香,キッシー岸田：CDで覚えるやさしい楽譜の読み方,(成美堂出版, 2005年).
[20] 野口 修義：作曲本 ～メロディーが歌になる～,(シンコーミュージック・エン

ターテイメント， 2005年).
[21] 辻 志朗：誰でもぜったい楽譜が読める！,(音楽之友社， 2005年).
[22] 江口 寿子：スケールは音楽の扉を開ける　スケールっておもしろい,(全音楽譜出版社， 2000年).
[23] アルノルト・シェーンベルク：作曲の基礎技法,(音楽之友社， 1971年).
[24] 金子 卓郎：コード進行による作曲入門ゼミ,(自由現代社， 2004年).

付録　ソースコード

付録 1．ユーザの嗜好情報をまとめた CSV ファイルから度数分布表を作成し，コード進行を出力する C プログラム

(c)2009 kiguchi

```
// csvファイルから度数分布表を作成し，コード進行をmidi形式で出力するプログラム
// (c)2009 kiguchi

#include "maedadata.h"
#include "maedamml.h"
#include <stdlib.h>
#include <string.h>
#include <time.h>

enum{
chordN=7   //chordN:使用するコードの数
};


//ランダム関数
int GetRandom(int min,int max)
{
  return min+(int)(rand()*(max-min+1.0)/(1.0+RAND_MAX));
}


//度数分布表から次に遷移するコードを決定する関数
int Getchord( int **output,int nowchord)
{
   int sum=0;
  for(int i=0;i<7;i++)
  {
          sum+=output[nowchord][i];
  }

  int x=GetRandom(0,sum);
  int y=0;

  for (int i=0;i<7;i++)
  {
          y+=output[nowchord][i];
          if(x<y) return i;

  }

  return 6;
}


//状態遷移のための度数分布表を作る関数
void joutaisenni(const char* filename,int** output)
{
    FILE *fp;
    char str[64];

    int f,s,t;
    char table[][4]={
          {"C_"},
        {"Dm7"},
          {"Em7"},
          {"F_"},
          {"G7_"},
          {"Am7"},
          {"Bm7"}};

    fp=fopen(filename,"r");
    if(!(fp=fopen(filename,"r"))){
        printf("読み込み失敗¥n");
        exit(1);
    }

    for(int i=0;i<chordN;i++){
       for(int j=0;j<chordN;j++)
          output[i][j]=0;}              //度数分布表の初期化

    for(;;){
      if(fgets(str,63,fp)==NULL)break;
```

```
      if(str[10]=='1'){
          for(int j=0;j<chordN;j++)
        {
                    if( strncmp(table[j],str,3)==0)   f=j;            //f=First 最初のコード
                    if( strncmp(table[j],str+3,3)==0) s=j;            //s=Second 2番目のコード
                    if( strncmp(table[j],str+6,3)==0) t=j;            //t=Third  3番目のコード

        }
          output[f][s]+=1;    //度数分布表のf行s列に+1
          output[s][t]+=1;    //度数分布表のs行t列に+1
      }
    }
    fclose(fp);
}


// メイン関数
int main(int argc,char* argv[])
{
    int nowchord,endchord,sum;

    int** output;             //度数分布表の作成
    srand(time(NULL));
    output = (int **)malloc(chordN *sizeof(int*));
    output[ 0 ] = ( int* )malloc( sizeof(int) * chordN * chordN);
    for( int i = 1; i < chordN; ++i ) output[ i ] = output[ 0 ] + chordN * i;
    joutaisenni(argv[1],output);

    MAEDA::Maeda maeda;

    int oct = 4;        // オクターブ

    maeda.set_scale( SCALE_C );                   // 音階
    maeda.set_tempo( 100 );             // テンポ
    maeda.set_prog( 1 );                // 音色 ( GMのプログラム番号 )
    maeda.set_prog_code( 2 );           // コードの音色 ( GMのプログラム番号 )
    maeda.set_velocity( 100 );          // 音量
    maeda.set_velocity_code( 80 );      // コードの音量

   //最初のコード決定
   switch(GetRandom(0,2))
   {
          case 0:{
                    maeda.set_code( CODE_C );     // コード
                    maeda.set_rest( LNG_2 );      // 休符
                    nowchord=0;
                    endchord=0;
                    break;
                    }

          case 1:{
                    maeda.set_code( CODE_Em7 );   // コード
                    maeda.set_rest( LNG_2 );      // 休符
                    nowchord=2;
                    endchord=2;
                    break;
                    }

          case 2:{
                    maeda.set_code( CODE_Am7 );   // コード
                    maeda.set_rest( LNG_2 );      // 休符
                    nowchord=5;
                    endchord=5;
                    break;
                    }

   }


   for(int senni=0;senni<5;senni++)
   {
        int chord=Getchord( output,nowchord);     //度数分布表から次に遷移するコードを決定する
        printf("%d¥n",chord);
          switch(chord)
          {
                    case 0:{
                              maeda.set_code( CODE_C );     // コード
                              maeda.set_rest( LNG_2 );      // 休符
                              nowchord=0;
                              break;
                              }

                    case 1:{
                              maeda.set_code( CODE_Dm7 );   // コード
                              maeda.set_rest( LNG_2 );      // 休符
                              nowchord=1;
                              break;
                              }

                    case 2:{
                              maeda.set_code( CODE_Em7 );   // コード
                              maeda.set_rest( LNG_2 );      // 休符
                              nowchord=2;
                              break;
```

```
                        }

                case 3:{
                        maeda.set_code( CODE_F );       // コード
                        maeda.set_rest( LNG_2 );        // 休符
                        nowchord=3;
                        break;
                        }

                case 4:{
                        maeda.set_code( CODE_G7 );      // コード
                        maeda.set_rest( LNG_2 );        // 休符
                        nowchord=4;
                        break;
                        }

                case 5:{
                        maeda.set_code( CODE_Am7 );     // コード
                        maeda.set_rest( LNG_2 );        // 休符
                        nowchord=5;
                        break;
                        }

                case 6:{
                        maeda.set_code( CODE_Bm );      // コード
                        maeda.set_rest( LNG_2 );        // 休符
                        nowchord=6;
                        break;
                        }
        }
    }


    //最後のコード決定（最初のコードと同じコードにする）
    switch(endchord)
    {
        case 0:{
                maeda.set_code( CODE_C );       // コード
                maeda.set_rest( LNG_2 );        // 休符
                break;
                }

        case 2:{
                maeda.set_code( CODE_Em7 );     // コード
                maeda.set_rest( LNG_2 );        // 休符
                break;
                }

        case 5:{
                maeda.set_code( CODE_Am7 );     // コード
                maeda.set_rest( LNG_2 );        // 休符
                break;
                }
    }

     maeda.save_smf( argv[2] ); // SMF ファイル出力

     free(output[0]);
     free(output);

     return 0;
}
```

付録 2. ユーザネーム登録画面の PHP ソース

(c)2008-2009 kanomata,kiguchi

```
<?php

//もし 1 曲目(初回表示)なら, result.csv へ曲名の配列だけ出力.
if($_GET['nowsongNO'] == 0)
{
  //songname に曲名を入れる
  $songname = file("temp.csv");

  //temp.csv から全楽曲数$allsongNO を取得.
  $allsongNO = count($songname);

  //result.csv に曲名のみを出力.
  $fp = fopen(($_GET['username']."_result.csv"),"w");

  echo($_GET['username']."_result.csv");
  for($i = 0; $i < $allsongNO; $i++)
  {
    fputs($fp,$songname[$i]);
  }
  fclose($fp);

 for($i = 0; $i < $allsongNO; $i++)
  {
    $order[$i] = $i;
  }
  shuffle($order);

  //username_order.csv に曲順を出力
  $fp = fopen(($_GET['username']."_order.csv"),"w");
  for($i = 0; $i < $allsongNO; $i++)
  {
    fputs($fp,$order[$i]);
    fputs($fp,"¥n");
  }
  fclose($fp);
}


//現在の曲名を取得.
$data_order = file(($_GET['username']."_order.csv"));
$num = trim($data_order[$_GET['nowsongNO']]); //order.csv から nowsongNO 番目に処理する楽曲番号$num を取得.
$data_result = file(($_GET['username']."_result.csv"));
$nowsongname = trim($data_result[$num]); //num 番目の楽曲名を現在の曲名$nowsongname として取得.
$allsongNO = count($data_result);


//次のページのアドレス設定. 全曲の印象入力が終わったら end.php へ. それ以外は本ページへリンク.
if( $allsongNO == $_GET['nowsongNO']+1 )
{
  $nextadd="end.php";
}
else
{
  $nextadd="input.php";
}


//印象を result.csv へ保存する.
if($_GET['nowsongNO'] != 0)
{
  $data_result = file(($_GET['username']."_result.csv"));
  $data_result[$_GET['num']] = (trim($data_result[$_GET['num']]).",".$_GET['impress']."¥n");
  $fp = fopen(($_GET['username']."_result.csv"),"w+");
  for($i = 0; $i < $_GET['allsongNO']; $i++)
  {
    fputs($fp,$data_result[$i]);
  }

  fclose($fp);
}

?>


<html><body>

<form action=<?php echo $nextadd; ?> method="get">

<div align="right">
全<input type="text" size=2 name="allsongNO" value="<?php echo $allsongNO; ?>">曲中
<br>
現在<input type="text" size=2 name="nowsongNO" value="<?php echo $_GET['nowsongNO']+1; ?>">曲目 <!--現曲数に 1 を足してボックスへ.
```

```
次$_GET['nowsongNO']. -->
</div>

<center>

<!-- コード進行パターンの swf ファイル出力 -->
<embed src="<?php echo("./data/".$nowsongname.".swf"); ?>"
       type="application/x-shockwave-flash"
       width="150" height="40"
       autostart="true"
       loop="true" repeat="true">
</embed>

<br>
曲名<input type="text" name="songname" value="<?php echo $nowsongname; ?>">
<br>
<br>

好き<input type="radio" name="impress" value="1"> <!--次$_GET['impress']. -->

嫌い<input type="radio" name="impress" value="0">

<br>
<br>

<input type="hidden" name="username" value="<?php echo $_GET['username']; ?>">
<input type="hidden" name="num" value="<?php echo $num; ?>">

<input type="submit" value="次の曲へ" >
<br>
<br>
<br>
※「次の曲へ」のクリックは, ENTERで代用可.

<a href="./index.php"> トップに戻る </a>

</center>

</form>

</body></html>
```

付録 3. 嗜好情報入力画面の PHP ソース

(c)2008-2009 kanomata,kiguchi

```
<?php

//もし 1 曲目(初回表示)なら, result.csv へ曲名の配列だけ出力.
if($_GET['nowsongNO'] == 0)
{
  //songname に曲名を入れる
  $songname = file("temp.csv");

  //temp.csv から全楽曲数$allsongNO を取得.
  $allsongNO = count($songname);

  //result.csv に曲名のみを出力.
  $fp = fopen(($_GET['username']."_result.csv"),"w");

  echo($_GET['username']."_result.csv");
  for($i = 0; $i < $allsongNO; $i++)
  {
    fputs($fp,$songname[$i]);
  }
  fclose($fp);

 for($i = 0; $i < $allsongNO; $i++)
  {
    $order[$i] = $i;
  }
  shuffle($order);

  //username_order.csv に曲順を出力
  $fp = fopen(($_GET['username']."_order.csv"),"w");
  for($i = 0; $i < $allsongNO; $i++)
  {
    fputs($fp,$order[$i]);
    fputs($fp,"¥n");
  }
  fclose($fp);
}


//現在の曲名を取得.
$data_order = file(($_GET['username']."_order.csv"));
$num = trim($data_order[$_GET['nowsongNO']]); //order.csv から nowsongNO 番目に処理する楽曲番号$num を取得.
$data_result = file(($_GET['username']."_result.csv"));
$nowsongname = trim($data_result[$num]); //num 番目の楽曲名を現在の曲名$nowsongname として取得.
$allsongNO = count($data_result);


//次のページのアドレス設定. 全曲の印象入力が終わったら end.php へ. それ以外は本ページへリンク.
if( $allsongNO == $_GET['nowsongNO']+1 )
{
  $nextadd="end.php";
}
else
{
  $nextadd="input.php";
}


//印象を result.csv へ保存する.
if($_GET['nowsongNO'] != 0)
{
  $data_result = file(($_GET['username']."_result.csv"));
  $data_result[$_GET['num']] = (trim($data_result[$_GET['num']]).",".$_GET['impress']."¥n");
  $fp = fopen(($_GET['username']."_result.csv"),"w+");
  for($i = 0; $i < $_GET['allsongNO']; $i++)
  {
    fputs($fp,$data_result[$i]);
  }

  fclose($fp);
}

?>


<html><body>

<form action=<?php echo $nextadd; ?> method="get">

<div align="right">
全<input type="text" size=2 name="allsongNO" value="<?php echo $allsongNO; ?>">曲中
<br>
現在<input type="text" size=2 name="nowsongNO" value="<?php echo $_GET['nowsongNO']+1; ?>">曲目 <!--現曲数に 1 を足してボックスへ.
```

```
次$_GET['nowsongNO']. -->
</div>

<center>

<!-- コード進行パターンの swf ファイル出力 -->
<embed src="<?php echo("./data/".$nowsongname.".swf"); ?>"
       type="application/x-shockwave-flash"
       width="150" height="40"
       autostart="true"
       loop="true" repeat="true">
</embed>

<br>
曲名<input type="text" name="songname" value="<?php echo $nowsongname; ?>">
<br>
<br>

好き<input type="radio" name="impress" value="1"> <!--次$_GET['impress']. -->

嫌い<input type="radio" name="impress" value="0">

<br>
<br>

<input type="hidden" name="username" value="<?php echo $_GET['username']; ?>">
<input type="hidden" name="num" value="<?php echo $num; ?>">

<input type="submit" value="次の曲へ" >
<br>
<br>
<br>
※「次の曲へ」のクリックは, ENTERで代用可.

<a href="./index.php"> トップに戻る </a>

</center>

</form>

</body></html>
```

付録 4. 終了画面の PHP ソース

(c)2008-2009 kanomata,kiguchi

```
<?php

//最終曲の印象を csv へ出力.

 $csv = ($_GET['username']."_result.csv");
 $mid = ($_GET['username']."_result.mid");
 $data_result = file(($_GET['username']."_result.csv"));
 $data_result[$_GET['num']] = (trim($data_result[$_GET['num']]).",".$_GET['impress']."¥n");
 $fp = fopen( $csv,"w+");
 for($i = 0; $i < $_GET['allsongNO']; $i++)
 {
   fputs($fp,$data_result[$i]);
 }
fclose($fp);

//csv ファイルから.mid ファイルを作成するプログラムを実行
$ret = exec( "/home/kiguchi/public_html/cgi-bin/maeda/maeda ".$csv." ".$mid, $out );

//作成された.mid ファイルを再生
print( "<html><body>ended!" );
print( "<EMBED SRC='$mid' HEIGHT='2' WIDTH='2' AUTOSTART='TRUE' REPEAT='FALSE' LOOP='FALSE' TYPE='audio/midi'>" );
print( "</body></html>" );
?>
```