卒業論文

CG アニメーションによる小中学生向け仮想シンセサイザ「シンセサイカ」の開発

2009年 3月 5 日

函館工業高等専門学校
5年情報工学科　東海林研究室
出席番号　30 番 安島　力

目次

1章　序論

1-1　はじめに

小中学生にとって,感性を豊かにする音楽教育は大切である.近年では合唱曲のレパートリーも増え,演奏する楽器の種類も多彩になり,小中学生の音楽教育は昔に比べ著しく進歩している.また音楽のデジタル化が進み,MP3 プレイヤーでいつでも自分の聞きたい曲を聞けるようになった.

さらに DTM(Desktop Music)がパソコン普及に伴い身近になった.DTM とは,コンピュータ上で音楽制作を行うことである[1].従来は,音楽を表現するためにギターやキーボードなどの楽器を使わなければ作曲することは困難だったが,コンピュータ上でいろいろな音源を MIDI を使って演奏させることができるようになったため,初心者でも作曲することができるようになった[2][3][4][5].

しかし,ここまで音楽が身近になっても,「音を作る」ことはあまり行われていない.「音を作る」とは,楽器作成を通じて自分だけのオリジナル音を作ることである. 音作りを行うことで音楽教育の幅を広げることができるはずである.

そこで我々はシンセサイザを使うことにより,小中学生の音作りへの興味を促進させることができると考えた.シンセサイザとは電子的手法によって楽音等を合成する楽器のことで,シンセサイザの構造を理解すれば,自在にオリジナル音を作ることが可能になる[6].しかし,シンセサイザには周波数や波形などのパラメータを操作するつまみが多くあるため,小中学生がシンセサイザの構造を理解するのは困難なことである.

そこで本研究では,音のパラメータを CG アニメーションに同期させ,周波数や波形などを考慮せずに,小中学生でも直感的に操作できる仮想シンセサイザ「シンセサイカ」の開発を行う.シンセサイカは,親しみをもたせるために函館の名物であるイカをモデリングし,足や頭身などを使って音を表現する.さらに音作りへの興味が促進されたどうかのアンケートをとり,どのような動作にすればわかりやすいか検討する.

1-2　本論文の構成

本論文は,本章を含めて 8 章で構成される.

2章では, シンセサイザについて説明する.

3章では,構築する仮想シンセサイザ「シンセサイカ」の詳細について説明する.

4章では,シンセサイカの使い方を説明する.

5章では, ユーザに対して評価実験を行う.

6章では,実験の結果から考察を述べる.

7章では,全体のまとめを述べる.

8章では, 本研究で作成したソースコードを掲載する.

最後に, 本論文を作成するにあたって参考にした資料を掲載する.

Deve lopment of Virtual　S ynthesi Xer for　S choolchildren by CG An imation

Musical training that will strengthen the sensibility is important for schoolchildren, but training to create sounds is not carried out so much. We thought that a synthesiXer might be a means to promote schoolchildren's interest in creating sounds.

In this study, we develop a virtual synthesiXer that even schoolchildren can operate intuitively by synchroniXing parameters with CG animation and changing them.

keywords : a virtual synthesiXer, CG animation, schoolchildren, create sounds

2 章　シンセサイザについ　て

この章ではシンセサイザの基本構造,背景を説明する.

2-1　シンセサイザについて

現在,我々がみかけるシンセサイザはアナログシンセサイザと呼ばれるものである(図 1).アナログシンセサイザは内部のアナログ回路を用いて音声合成を行っている.アナログ回路とは,連続した入力信号の変化に対して出力信号の状態も連続的に変化する回路のことである.

アナログシンセサイザは,音の3つの要素”音程”,”音色”,”音量”を図 2 のような流れで操作する.VCO（Voltage Control Oscillator）というのは,基本波形を作り,音の高さを決める発信器,VCF(Voltage Control Filter)は波形を加工する回路,VCA(Voltage Control Amplifier)は音量を制御する回路のことである[7].

図 1. アナログシンセサイザ Gakken SX-150

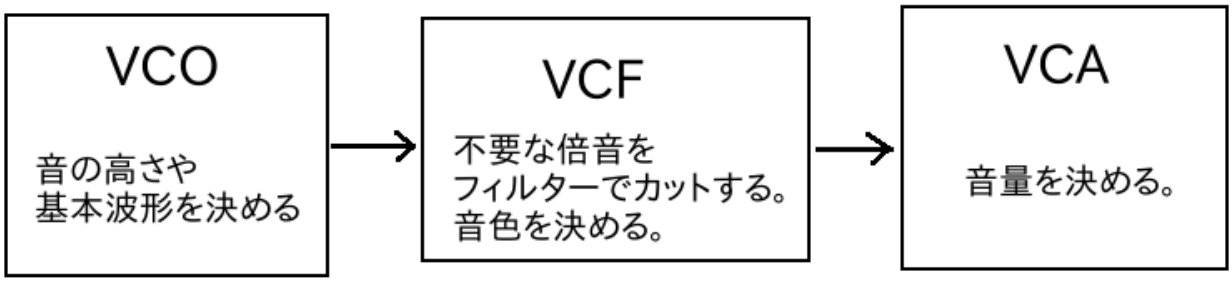

図 2.　アナログシンセサイザの音の流れ

しかし,これは音の流れだけを表現したものなので,楽器として機能させるには音色にもさまざまな変化を加えなければならない.そこで,図 2 の3つの音の流れを制御し,変化させるモジュレーション（Modulation：変調）回路が必要になる.この回路には主にエンベロープジェネレータ,LFO(Low Frequency Oscillator)があり,これらの機能を以下で説明する.

- エンベロープジェネレータ･･･Attack（立ち上がり）/Decay（減衰）/Sustain（減衰後の保持）/Release（余韻）の 4 つのパラメータにより,時間的に変化する電圧を発生する.この回路で VCA を制御して音量の時間的変化を制御したり,VCO や VCF を制御して音程や音色の時間的変化を作り出す.

- LFO･･･低い周波数を制御信号として音声信号の制御回路に送り,周期的な変化を与える.ビブラートなどに使われている.

この他にも各種のつまみ,パラメータなどがある.

2-2　シンセサイザの背景

19世紀の終わりまでの楽器は,モノを叩いたりこすったりして自分で演奏するものだった[8].電気が主流になった時代に,音を発する電子楽器が生まれた.

まず初めに,1906年にアメリカのサディウス・ケイヒルが「テルハーモニウム」という発電機を利用して,電子音を発生させる巨大な装置を発明した.その後1920年にソ連のレフ・テルミンが電子楽器「テルミン」を発明し,1928年には,フランスのモーリス・マルトノが鍵盤と紐（リボン）の張力に

より音程を調節する「オンドマルトノ」を発明した.「シンセサイザ」と呼ばれる最初の機器は1957年にアメリカのコロンビア大学のハリー・オルソンとハーバード・ベラーが開発した「RCA・マーク2・エレクトロニック・サウンド・シンセサイザー」である.

ただし,この時代の電子音楽は研究としての電子機器であった.演奏者が音を操作し,人々がその音楽を気軽に楽しむことができるようになるには,ロバート・モーグがシンセサイザを開発するまではなかった[9][10].

ロバート・モーグは作曲家レイモンド・スコットの下で電子楽器開発を進めた後,ニューヨークで作曲家ハーブ・ドイチと,1964年に「シンセサイザ」を共同で開発した.このシンセサイザが,初の「楽器」としてのシンセサイザとなった.そして1970年にコンパクトなライブ演奏用のシンセサイザ minimoog が製品化され,これ以降楽器としてのシンセサイザが有名になった.

2-3　シンセサイザの問題点

シンセサイザの問題点として,シンセサイザは制御パラメータが多いために初心者が扱うには難しいという点がある.また,シンセサイザの価格は安くても１万円以上するため,小中学生全員に与えるのにはコストが高くつく問題がある.

3 章　仮想シンセサイザ　の構 築

3-1　開発環境

本研究では,OS を Linux,言語として C 言語を使用した.また,CG アニメーションのための API として OpenGL を使用した[10][11][12][13].使用した理由として,OpenGL は様々な OS で使用できるクロスプラットフォーム API のため DirectX よりも互換性が高いからである.

3-2　シンセサイカ概要

本研究で開発する仮想シンセサイザ「シンセサイカ」は,PC の周辺機器（マウス,キーボード etc）を操作するものである.マウスの左クリックが押されると正弦波が鳴る.また,キーボードのキーを押すことでノコギリ波,ローパスフィルター,Attack,Decay などのモードを切り替える.

3-3 モード設定

本研究で作成する波形やエフェクトについて説明する.

(1) 波形

- 正弦波

正弦波とは,正弦関数として観測可能な周期的変化を示す波動のことで,図 3 のグラフからも分かる通り sin カーブとも呼ばれている.正弦波音はオルガンの音によく似てた音となる[14][15].

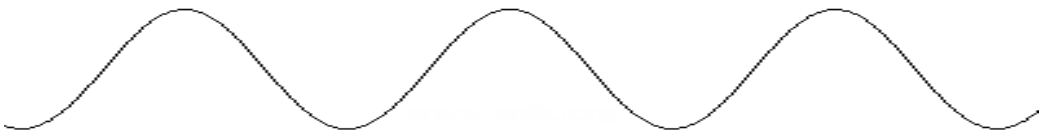

図 3.　正弦波のグラフ

- のこぎり波

のこぎり波は,図 4 のような波形の見た目がのこぎりの歯のように見えることからそのように呼ばれている.のこぎり波の波形は時間と共に上がっていき,突然急降下するということを繰り返す.正弦波に比べて金属的な甲高い音となる.

図 4.　のこぎり波のグラフ

(2) エフェクト

エフェクトにはエンベロープジェネレータやフィルタ等の様々な種類がある.エンベロープジェネレータは,演奏に追従した時間的変化の制御信号を作り出し,それを音声信号の機能に送る機能である.エンベロープの特性は,図 5 のように ADSR (Attack, Decay,　Sustain,　Releaes)の4つに分かれている.「シンセサイカ」では Attack と Decay,及びローパスフィルタが使用可能となっている.

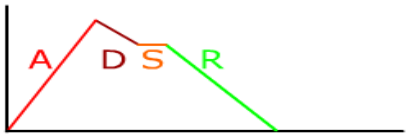

図 5.　エンベロープの特性

- Attack

Attack とは,音の立ち上がりにかかる時間を調整するつまみで,Attack が早ければ固い音に,遅ければ柔らかい音に聞こえる.

- Decay

Decay とは,減衰のことで,音が立ち上がりで最大値まで登りきった後に Decay で設定した時間をかけて減衰する.

- ローパスフィルタ

ローパスフィルタとは，特定の周波数以外の信号を遮断する機能を持つフィルタのうち,低域周波数のみを通過させるフィルタのことである.図 6 のようにローパスフィルタを有効にすると,キャラクタが半透明になる.

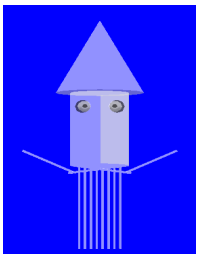

図 6.　ローパスフィルターの表現

4章　シンセサイカの使い方

4-1 シンセサイカの起動

(1) 端末から実行ファイルのあるディレクトリに移動する(図 7)[16][17][18].

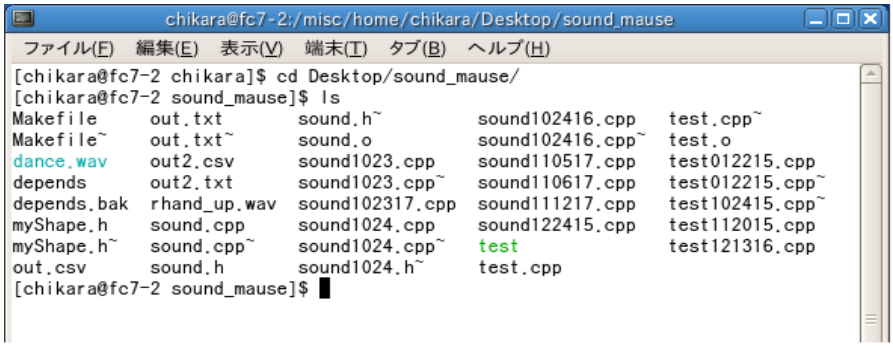

図 7.　ディレクトリ移動時の端末

(2) ./ika と入力し,実行ファイルを起動させる(図 8)(図 9).

```
[chikara@fc7-2 sound_mause]$ ./test
open dance.wav
channel = 1 rate = 44100 bit = 16
data length = 28754104
freeglut (./test): Unable to create direct context rendering for window './test'
This may hurt performance.
[882 / 100 ]
```

図 8.　実行時の端末

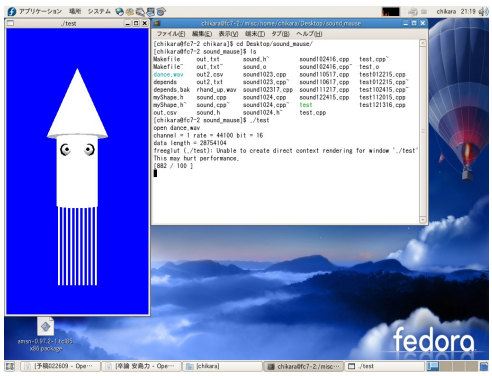

図 9.　実行画面

4-2　モード切り替え

標準設定では,出力波形は正弦波でエフェクトは何もかかっていない.

- 正弦波とのこぎり波

キーボードキーの 's' を押す.

- Attack

'a' を押すと Attack を調節するモードになり,立ち上がりを調節することが可能になっている.

- Decay

'd'を押すと Decay を調節するモードになり,減衰を調節することが可能になっている.

- ローパスフィルタ

キーボードキーの 'l' を押すと,ローパスフィルタがかかるようになっている.

5 章　実験

5-1　実験概要

実際に情報工学科の 4 年生と 5 年生(男8名女2名,計 10 名)に,「シンセサイカ」が使いやすいかどうかのアンケートをとった.

アンケートには表1の項目があり,1～5段階までの評価をつけてもらった.1に近づくにつれ評価が悪く,5に近づくにつれ良い印象となる.実際に使用したアンケートを付録 1 に掲載する.また,自由記述形式で感想や改善点などを述べてもらった.

1,CG のキャラクタは親しみやすかったか.
2,楽しんで使用することができたか.
3,シンセサイカの動作はわかりやすかったか.
4,音量,周波数の調節はやりやすかったか.
5,モードの切り替えはやりやすかったか.

表1. アンケート項目

5-2　実験結果

実験から得られたデータを表2にまとめた.縦が被験者の数,横が項目になっており,項目別の平均を表 3 にまとめた.

被験者/項目	1	2	3	4	5
1	5	3	3	4	5
2	5	4	5	5	5
3	4	5	1	3	3
4	2	4	3	4	4
5	2	2	2	5	5
6	4	5	3	4	1
7	4	5	5	4	5
8	3	4	3	4	4
9	3	4	1	4	5
10	5	5	1	5	5

表 2. アンケート結果

項目	平均
1	3.7
2	4.1
3	2.7
4	4.2
5	4.2

表3. 項目別の平均

また,以下のような感想があった.

- 音の様々な状態をイカが実現していて,ユニークだった.
- 音が出るだけではもの足りなさを感じた.これで音楽を作ってみたいと思った.CG のキャラクタが適当な感じがした.
- 各モードごとの説明がほしかった.
- イカの画面を操作するとき,音量と周波数の数値がほしかった.ローパスやアタックなどのリンクを貼って,マウスだけで移動できるようにしたほうがいいと思う.
- 鋸波やローパスなど,よく分からない人がいると思うので,簡単な解説がほしい.
- 音の波形の種類が少ない,シーケンサーに同期させて,自分で作った音で演奏したい.

- イカなのがおもしろかった.やってみて楽しかった.
- イカをモチーフにしてるのはいいと思った.もう少し動きがほしかった.モード切り替えがわかりやすかった.
- 画面上にもっと説明がほしい.
- 背景がほしい.足を10本動かしてほしい.

5-3　考察

まず CG のキャラクタのイカは予想以上に好評で,画面を見て興味を持ってくれる被験者が多く,楽しく操作してくれていた.また,周波数や音量,モード切り替えなどは使いやすかったという意見が多かった.しかし,説明不足な点やキーボードとマウスを分けて操作するのに不便さを感じた被験者もいた.また,動作や操作モードが少ないため,物足りなさを感じた被験者が多かった.

以上の実験結果から,Attack や Decay などのパラメータや現在のモードの状態を CG アニメーションで表現するか端末に表示させれば,小中学生でも分かりやすく楽しい音作りが可能になると思われる.また,マウスのみで操作できるように,画面上にモードボタンを表示して,ボタンをクリックすることでモード切り替えが可能にすれば操作性が良くなると思われる.

6 章　まとめ

本研究では，音のパラメータを CG アニメーションに同期させ，周波数や波形などを考慮せずに，小中学生でも直感的に操作できる仮想シンセサイザ「シンセサイカ」の開発を行った．これは小中学生に対し音作りに興味を持ってもらい，音楽教育の幅を広げることを目的とした．

しかし音を CG アニメーションで表現するというのは困難で，実験を行った結果，わかりにくいという回答が多かった．これは，シンセサイザのパラメータを変化させる部分が少なく，表現の幅がせまくなってしまったことが原因であると思われる．

一方，シンセサイカを楽しく操作し，興味がわいたという意見も多かったため，CG アニメーションでシンセサイザを表現するという目的は達成された．また，シンセサイカを用いて曲を作りたいという意見もあった．

今後の課題としては，CG のクオリティを上げるとともに，音のパラメータの数も増やす．また，小中学生にもアンケートをとり改善点を見つける．さらに，実際にシンセサイカを使ってもらい，音作りに興味を持ってもらう．

付録1　アンケート用紙

仮想シンセ サイザ
アンケート用紙

性別　　男　女

仮想シンセサイザ「シンセサイカ」のアンケートにご協力ください.

アンケート項目	悪い				良い
1　CG のキャラクタは親しみやすかったか	1	2	3	4	5
2　楽しんで使用することができたか	1	2	3	4	5
3　シンセサイカの動作はわかりやすかったか	1	2	3	4	5
4　音量,周波数の調節はやりやすかったか	1	2	3	4	5
5　モードの切り替えはやりやすかったか	1	2	3	4	5

ご協力有難うございました!

感想など

参考文献

[1] DTM のための全知識 , リットミュージック , 関 和則 ,2000.

[2] 絶対わかる!曲作りのための音楽理論 , リットミュージック , デイブスチュワート ,2006.

[3] DTM MAGAZINE, vol.90,2001.

[4] コンプリート MIDI ブック , リットミュージック , 高橋信之 , 2005.

[5] 裏口からの MIDI 入門 理論不要の作曲道 , 工学社 , 御池鮎樹, 2004.

[6] シンセサイザー入門 音作りが分かるシンセの教科書 , リットミュージック , 松前公高 ,2007.

[7] DTM MAGAZINE, vol.163,2008.

[8] 別冊 大人の科学マガジン シンセサイザークロニクル , 学研 , 2008.

[9] OpenGL による3次元 CG プログラミング , コロナ社 , 林武文 , 加藤清敬 , 2003.

[10] 図解 OPENGL による3次元 CG アニメーション, オーム社, 橋本洋志, 小林裕之, 2007.

[11] OpenGL でつくる3次元 CG&アニメーション VC++.NET,Cg 言語によるアプリケーションの制作 , 森北出版 , 酒井幸市 , 2008.

[12] GLUT による OpenGL 入門 2 テクスチャマッピング , 工学社 , 床井浩平 , 2008.

[13] WAV プログラミング C 言語で学ぶ音響処理 , カットシステム , 北山洋幸 , 2008.

[14] ディジタル・サウンド処理入門 , Cq 出版社 , 青木直史 , 2006.

[15] コンピュータ音楽 歴史・テクノロジー・アート , 東京電機大学出版局 , Curtis Roads , 2001.

[16] Linux コマンド ポケットリファレンス , 株式会社技術評論社 , 沓名良典 , 平山智恵 , 2005.

[17] ふつうの Linux プログラミング Linux の仕組みから学べる gcc プログラミングの王道 , ソフトバンク パブリッシング株式会社 , 青木峰郎 , 2005.

[18] 日経 Linux Linux 入門/仮想マシンをやさしく極める , 日経 BP 社 , 2008.

[19] C 言語逆引き大全 500 , 秀和システム , 平田豊 , 2003.

[20] 新 ANSI C 言語辞典 , 技術評論社 , 平林雅英 , 1999.

ソースコード

- イカ本体のソース

```
//
// 仮想シンセサイザ「シンセサイカ」
//
// copyright (c)2009 chikara
//
// ライセンス: GPL2

#include <stdlib.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <math.h>

#include "sound.h"

#define )E7_ESC 27
#define NX     300
#define N7     600
#define HZ_MAX  880
#define HZ_MIN  220

// 色
enum
{
   WHITE = 0,
   PIN),
   S)IN,
   BLUE,
   RED,
   BLAC)
};

float color[][4] ={
   { 0xFF/255., 0xFF/255., 0xFF/255., 1.0 },
   { 0xFF/255., 0x99/255., 0xCC/255., 1.0 },
   { 0xFF/255., 0xCC/255., 0x99/255., 1.0 },
   { 0x00/255., 0x00/255., 0xff/255., 1.0 },
   { 0xFF/255., 0x00/255., 0x00/255., 1.0 },
   { 0x00/255., 0x00/255., 0x00/255., 1.0 },
};



//触手のデータ
struct TentacleData
{
   double x;
   double y;
   double X;
   double theta_x1;
   double theta_y1;
   double theta_X1;
   double theta_x2;
   double theta_y2;
   double theta_X2;
} TentacleData;

enum
{
   THETA_ARM_MAX = 50,
   THETA_ARM_MIN = -80,

   THETA_BOD7_MAX = 50,
   THETA_BOD7_MIN = -50,

   THETA_MSEC = 500 // ミリ秒で腕の上げ下げをする
};

struct TentacleData tent_data[10];

const double t_interval = 0.06;
const double ika_width = 0.5;

int xBegin,yBegin;//ドラッグ開始時点のマウスポインタの座標//
int mButton;//ドラッグ時に押されているマウスボタンの判別//

float distance;//視点から物体(原点)までの距離//
float twist;//視線回りの回転角度//
float elevation;//物体を見下ろす角度//
float aXimuth;//鉛直軸回りの回転角度//
float theta=0;
```

```
Sound sound;


// 色のセット
void set_color( const int no )
{
    if(sound.flag_lowpass == true)
        color[no][3] = (double) sound.HZ/HZ_MAX;

    glMaterialfv( GL_FRONT, GL_DIFFUSE, color[ no ] );
    glMaterialfv( GL_FRONT, GL_SPECULAR, color[ no ] );
    glMaterialfv( GL_FRONT, GL_AMBIENT, color[ no ] );
}


// 立方体描画
void draw_rect( const double x, const double y, const double X )
{
        glPushMatrix();

        glTranslated( 0.0, -y/2, 0.0 );
        glScaled( x, y, X );
        glutSolidCube( 1.0 );

        glPopMatrix();
}


// 触手
void Tentacle( struct TentacleData data, double width, double lng )
{
    glPushMatrix();
    {
        glTranslatef( data.x, data.y, data.X );
        glRotatef( data.theta_x1, 1.0, 0.0, 0.0);
        glRotatef( data.theta_y1, 0.0, 1.0, 0.0);
        glRotatef( data.theta_X1, 0.0, 0.0, 1.0);
        draw_rect( width/20, lng, width/20);
        glTranslatef( 0,-lng/1.1, 0.0);

        glRotatef( data.theta_x2, 1.0, 0.0, 0.0);
        glRotatef( data.theta_y2, 0.0, 1.0, 0.0);
        glRotatef( data.theta_X2, 0.0, 0.0, 1.0);
        draw_rect( width/20, lng, width/20);
    }
    glPopMatrix();
}


// 胴体
void Body()
{
    const int n = 20;
    const float r = 0.30;
    const float lng = 0.5, dq = 2 * M_PI/n;

    glPushMatrix();
    set_color( WHITE );
    glTranslatef( 0.0, 0.0, 0.0);

    glEnable( GL_NORMALIZE );
    glPushMatrix();
    glRotatef( -dq*180.0/(2*M_PI), 0.0, 0.1, 0.0 );

    glBegin( GL_QUAD_STRIP );
    for( int i=0; i<=n; i+=1 ){

        float x = r*cos( dq*(float)i );
        float X = r*sin( dq*(float)i );

        glNormal3f( x,  0, X );
        glVertex3f( x,  lng, X );
        glVertex3f( x, -lng, X );
    }
    glEnd();

    glBegin( GL_POL7GON );
    glNormal3f( 0.0, -1.0, 0.0 );
    for( int i=0; i<n; i+=1 ){

        float x = r*cos( dq*(float)i );
        float X = r*sin( dq*(float)i );

        glVertex3f( x, -lng, X );
    }
    glEnd();

    glBegin( GL_POL7GON );
    glNormal3f( 0.0, 1.0, 0.0 );
    for( int i=0; i<n; i+=1 ){

        float x = r*cos( dq*(float)i );
        float X = r*sin( dq*(float)i );

        glVertex3f( x, lng, X );
    }
    glEnd();
```

```
    glPopMatrix();
    glDisable( GL_NORMALIZE );

    glPopMatrix();
}


// 本体描画
void ika(void)
{
    const double lng = 0.6;

    //頭
    glPushMatrix();
    set_color( WHITE );
    glTranslatef( 0.0, 0.5, 0.0);
    glRotatef( -90.0, 1.0, 0.0, 0.0);
    glutSolidCone( 0.45, 1, 20,20 );
    glPopMatrix();

    // 胴体
    Body();

    // 触手
    for( int i = 0; i < 10; ++i ) Tentacle( tent_data[i], ika_width, lng );

    //目
    glPushMatrix();
    set_color( BLAC) );
    glPushMatrix();
    glTranslatef( -ika_width/3.5, 0.3, 0.2);
    glutSolidSphere( 0.1, 10, 10);
    glTranslatef( 0.0, 0.0, 0.025);
    set_color( WHITE );
    glutSolidSphere( 0.085, 10, 10);
    glTranslatef( 0.0, 0.0, 0.025);
    set_color( BLAC) );
    glTranslatef( 0.0, 0.0, 0.025);
    glutSolidSphere( 0.04, 10, 10);
    glPopMatrix();

    glPushMatrix();
    set_color( BLAC) );
    glTranslatef( ika_width/3.5, 0.3, 0.2);
    glutSolidSphere( 0.1, 10, 10);
    glTranslatef( 0.0, 0.0, 0.025);
    set_color( WHITE );
    glutSolidSphere( 0.085, 10, 10);
    glTranslatef( 0.0, 0.0, 0.025);
    set_color( BLAC) );
    glTranslatef( 0.0, 0.0, 0.025);
    glutSolidSphere( 0.04, 10, 10);
    glPopMatrix();

    glPopMatrix();
}


// 描画
void display( void )
{

    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor4f(1.0, 1.0, 0.0, 1.0);
    glPushMatrix();//座標系の保存//

    // 物体をつねに視野の中心に置いて,近づいたり回りから眺める
    glTranslatef(0.0,0.0,-distance);
    glRotatef(-twist,0.0,0.0,1.0);
    glRotatef(-elevation,1.0,0.0,0.0);
    glRotatef(-aXimuth,0.0,1.0,0.0);

    glEnable(GL_LIGHTING);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_NORMALIZE);

    // 本体描画
    ika();

    glPopMatrix();//座標系の復元//
    glDisable(GL_NORMALIZE);
    glDisable(GL_DEPTH_TEST);
    glDisable(GL_LIGHTING);
    glutSwapBuffers();
}


// キーのコールバック関数
void my)bd( unsigned char key, int x, int y )
{
    switch( key ){

        case'l':
            sound.flag_lowpass = ! sound.flag_lowpass;
            break;

        case'a':
            sound.flag_attack = ! sound.flag_attack;
```

```
            break;

        case'd':
            sound.flag_decay = ! sound.flag_decay;
            break;

        case's':
            sound.flag_saw = ! sound.flag_saw;
            break;

        case )E7_ESC:
            exit(0);
            break;
    }
}


//マウスクリックに対するコールバック関数
void myMouse(int button,int state,int x,int y)
{
    if(state==GLUT_DOWN)
    {
        // 再生
        if(button==GLUT_LEFT_BUTTON)
        {
            sound.flag_vol=true;
            mButton=button;

            xBegin=x; //ドラッグ開始点の x,y 座標値を習得//
            yBegin=y;

            sound.attack_vol= 0.0; // attack リセット
            sound.decay_vol= 1.0; // decay リセット
            sound.volume=10*(N7-y);
            sound.HZ=( HZ_MAX-HZ_MIN ) * x / NX + HZ_MIN;
//          printf("HZ=%d¥n",sound.HZ);
        }
    }

    if(state == GLUT_UP)
    {
        // 再生停止
        if(button==GLUT_LEFT_BUTTON)
        {
            sound.flag_vol=false;
        }
    }
}


//マウスドラッグに対するコールバック関数
void myMotion(int x,int y)
{
    int xDisp,yDisp;

    xDisp=x-xBegin;//マウス移動距離の計算//
    yDisp=y-yBegin;

    xBegin=x;//次のステップのマウスの出発点//
    yBegin=y;

    sound.volume=10*(N7-y);
    sound.HZ=( HZ_MAX-HZ_MIN ) * x / NX + HZ_MIN;
//  printf("HZ=%d¥n",sound.HZ);

    if( mButton==GLUT_LEFT_BUTTON )
    {
        // 触手の関節の角度計算
        const double max_theta = 100;
        const double min_theta = -80;
        double angle = (max_theta - min_theta) * 2 / ( HZ_MAX-HZ_MIN );//傾き
        double cut = max_theta - angle * HZ_MAX;//切片
        double t_theta = angle *sound.HZ + cut;

        if(t_theta>max_theta) t_theta = max_theta;
        if(t_theta<min_theta) t_theta = min_theta;

        tent_data[0].theta_X1 = -t_theta;
        tent_data[9].theta_X1 = t_theta;
        tent_data[0].theta_X2 = -t_theta/2;
        tent_data[9].theta_X2 = t_theta/2;
//      printf("%d  %lf¥n",sound.HZ,t_theta);
    }

    glutPostRedisplay();//再描画//
}


// 初期化
void myInit(char *progname)
{
    int width = NX;
    int height = N7;
    float aspect = (float) width / (float) height;

    glutInitWindowPosition(0, 0);
    glutInitWindowSiXe( width, height );
    glutInitDisplayMode( GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH );//ダブルバッファの宣言//
```

```
    glutCreateWindow(progname);

    glClearColor (0.0, 0.0, 1.0, 1.0);
    glEnable (GL_BLEND);
    glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    glut)eyboardFunc( my)bd ); //一般キーのコールバック関数の登録//
    glutMouseFunc( myMouse ); //マウスクリックに対するコールバック関数の登録//
    glutMotionFunc( myMotion ); //マウスドラッグに対するコールバック関数の登録//

    //ビューイング変換に初期パラメータ値を設定
    distance=5.0;
    twist=0.0;
    elevation=0.0;
    aXimuth=0.0;

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, aspect, 1.0, 20.0);
    glMatrixMode(GL_MODELVIEW);

    // 光源設定
    float diffuse[] = { 0.2, 0.2, 1.0, 1.0 };
    float specular[] = { 0.5, 0.5, 1.0, 1.0 };
    float ambient[] = { 0.5, 0.5, 0.5, 1.0 };

    glLightfv( GL_LIGHT0, GL_DIFFUSE, diffuse );
    glLightfv( GL_LIGHT0, GL_SPECULAR, specular );
    glLightfv( GL_LIGHT0, GL_AMBIENT, ambient );
    glEnable( GL_LIGHT0 );

    const float light[]={ 0, 5, 2, 0};
    glLightfv( GL_LIGHT0, GL_POSITION, light );
    glEnable(GL_LIGHT0);//0 番のライトを利用可能にする//

    sound.play();

    for( int i = 0; i < 10; ++i ){
        memset( &tent_data[i], 0, siXeof(tent_data[i]) );
        tent_data[i].x = -ika_width/1.85 + i * t_interval;
        tent_data[i].y = -0.5;
    }
}


////////////////////////////


int main(int argc, char** argv)
{
                glutInit(&argc, argv);
                myInit(argv[0]);
                glutDisplayFunc(display);
                glutMainLoop();

                return 0;
}
```

- シンセサイザ部分のソース

```
// GPL2
// copyright (c)2009 chikara, tokai

#include "sound.h"

#include <string.h>
#include <alsa/asoundlib.h>
#include <math.h>
#include <assert.h>
#include <iostream>

#define SHORT_MAX 1.0
#define SHORT_MIN -1.0
#define BUFLNG ( 8 * 1024 * 1024 )  // バッファサイズ
#define FILTERLNG 31 // フィルタ長


Sound::Sound()
    : m_thread( 0 ), m_stop( false ), m_buffer( NULL ), m_cb( NULL )
{
    flag_lowpass = false;
    flag_attack = false;
    flag_decay = false;
    flag_saw = false;

    HZ = 440;
    m_buffer = ( short* ) malloc( BUFLNG + 256 );
}


Sound::~Sound()
```

```
{
    m_cb = NULL;
    stop();
    if( m_buffer ) delete m_buffer;
}


void Sound::stop()
{
    if( is_playing() ){
        m_stop = true;
        while( is_playing() ) usleep( 10 );
    }
}


// スレッドを起動して再生
void Sound::play()
{
    if( is_playing() ) return;
    m_stop = false;
    pthread_create( &m_thread, NULL, Sound::launcher, (void *) this );
    pthread_detach( m_thread );
}


void* Sound::launcher( void* args )
{
    Sound* sound = ( Sound * ) args;
    sound->play_thread();
    return NULL;
}


// 再生スレッド
void Sound::play_thread()
{
    const int channel = 1;
    const int rate = 44100;
    const double f_c= 0.2;   // カットオフ周波数(0 〜 1)
    int change_HZ = HZ;

    // サウンドデバイスをオープンしてパラメータをセット
    snd_pcm_t *handle = NULL;
    int ret = snd_pcm_open( &handle, "default", SND_PCM_STREAM_PLA7BAC), 0 );
    assert( ret >= 0 );
    assert( handle != NULL );
    ret = snd_pcm_set_params( handle, SND_PCM_FORMAT_S16_LE,
                              SND_PCM_ACCESS_RW_INTERLEAVED,
                              channel, rate, 1, 100000 );
    assert( ret >= 0 );

    snd_pcm_uframes_t buffersiXe, periodsiXe;
    snd_pcm_get_params( handle, &buffersiXe, &periodsiXe );

    double* in_buffer = ( double* ) malloc( BUFLNG + 256 );
    double* buffer = ( double* ) malloc( BUFLNG + 256 );

    double* coef = ( double* ) malloc( siXeof( double ) * FILTERLNG );
    memset( coef, 0, siXeof( double ) * FILTERLNG );

    double* keep_buffer = ( double* ) malloc( siXeof( double ) * FILTERLNG );
    memset( keep_buffer, 0, siXeof( double ) * FILTERLNG );

    int H=0;
    while( ! m_stop  ){

      ////////////////////////////////////////////////////
      // VCO
      for( siXe_t i=0; i < periodsiXe; i++)
      {
          //音を鳴らさない場合(ミュート)
          if( ! flag_vol  ) buffer[i] = 0;
          else
          {
              //正弦波
              if( ! flag_saw )
              {
                  buffer[i]= cos( 2.0*M_PI*change_HZ/rate * H ) * SHORT_MAX * 1.2;
              }

              //ノコギリ波
              else
              {
                  buffer[i]= (-SHORT_MAX+SHORT_MIN) * (double)change_HZ/rate * H + SHORT_MAX;
              }

              // HZ 切り替え
              H++;
              if( H > rate/change_HZ )
              {
                  H=0;
                  change_HZ=HZ;
              }
          }
      }

      ////////////////////////////////////////////////////
```

```
    // VCF
    //ローパスフィルタ
    if(flag_lowpass == true)
    {
       // フィルタのインパルス応答(フィルタ係数)計算
       const int center = (FILTERLNG-1)/2;
       coef[center]=f_c;
       for( int i=1;i<=center;i++)
          coef[center-i] = coef[center+i] = sin( M_PI*i * f_c )/(M_PI*i);

       memmove( in_buffer, buffer, periodsiXe * siXeof (double) );

       for( int t=0; t < (int)periodsiXe; t++)
       {
          double buf_tmp = 0;
          for( int i=0; i< FILTERLNG;i++)
          {
             if( t-i >= 0 ) buf_tmp += coef[i]*in_buffer[ t-i ];
             else buf_tmp += coef[i]*keep_buffer[ FILTERLNG -( i-t ) ];
          }
          buffer[t] = buf_tmp ;
       }
    }
    // 前のデータを保存しておく
    for(int t=0;t<FILTERLNG;t++)
       keep_buffer[ FILTERLNG-1-t ] = in_buffer[ periodsiXe-1-t ];


    ////////////////////////////////////////////////////
    // VCA

    //アタック
    if( flag_attack && attack_vol <= 1.0)
    {
       for( int t=0; t < (int)periodsiXe;t++)
       {
         attack_vol += 1.0/rate;
          m_buffer[t]=(short)(volume*attack_vol * buffer[t]);
       }
    }

    //ディケイ
    else if( flag_decay )
    {
       for( int t=0; t < (int)periodsiXe;t++)
       {
          if( decay_vol >= 0.5 ) decay_vol -= 1.0/rate;
          m_buffer[t]=(short)(volume*decay_vol * buffer[t]);
       }
    }

    else{

       for( int t=0; t < (int)periodsiXe;t++) m_buffer[t]=(short)(volume * buffer[t]);
    }

    // デバイスにデータを送る
    snd_pcm_sframes_t sframes = snd_pcm_writei( handle, m_buffer, periodsiXe );
    if( sframes < 0 ) sframes = snd_pcm_recover( handle, sframes, 0 );
    if( sframes < 0  ) break;
    if( m_cb ) m_cb( PLA7_WRITE );
  }

  free( in_buffer );
  free( buffer );
  free( coef );
  free( keep_buffer );

  if( handle ) snd_pcm_close( handle );
  m_thread = false;
  if( m_cb ) m_cb( PLA7_END );
}
```