2010年度文理学部体験授業

「パソコンをシンセサイザーにしてみよう」

日本大学 文理学部 情報システム解析学科
北原　鉄朗
<kitahara@cssa.chs.nihon-u.ac.jp>
http://virgo.cssa.chs.nihon-u.ac.jp/~kitahara/

シンセサイザーって何?

シンセサイザーとは

楽器の音などを電子的に合成する装置

ヤマハ MOTIF XF6

http://www.yamahasynth.com/より

ローランド JUNO-STAGE

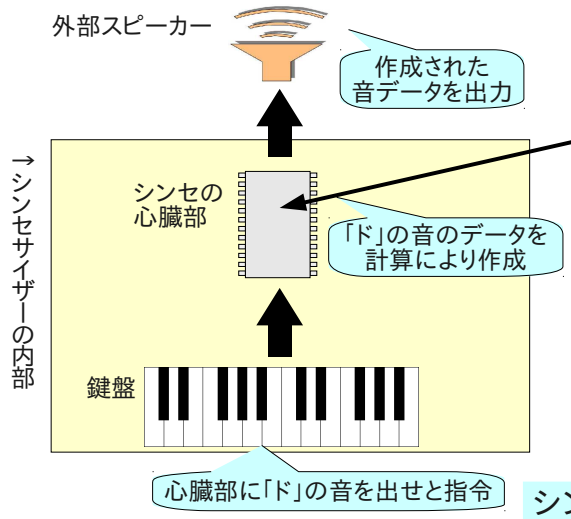
http://www.roland.co.jp/より

シンセサイザーの特徴

- 物理的な発音機構(弦や管など)を持たない (オルガンとの違い)
- ボタンやつまみの操作により、音色を様々に加工できる
- 鍵盤が内蔵されている場合が多い

シンセサイザーはコンピュータだ

シンセサイザーの内部構成

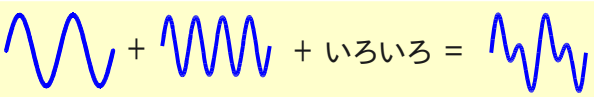


音データを作るための
小さなコンピュータ
（音データを作るための
プログラムが入っている）

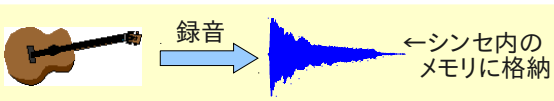
シンセサイザー内部の音を作る
コンピュータと同じものは、
通信カラオケでも使われている。

音を合成する3つの方式

- 古典的合成方式
  - 単純な波形を重ねたり加工することで音を作り出す

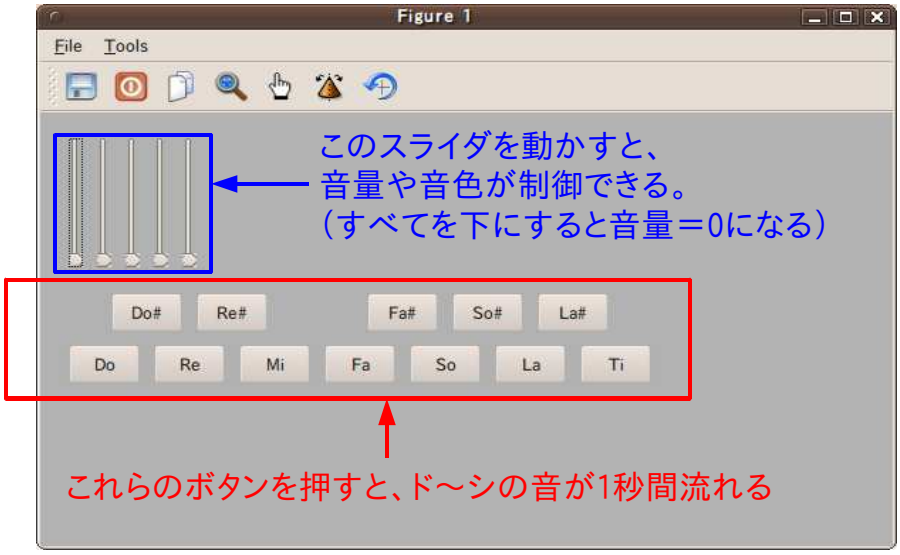


- PCM合成方式
  - 実際の楽器音をデジタル録音し、その波形を用いる



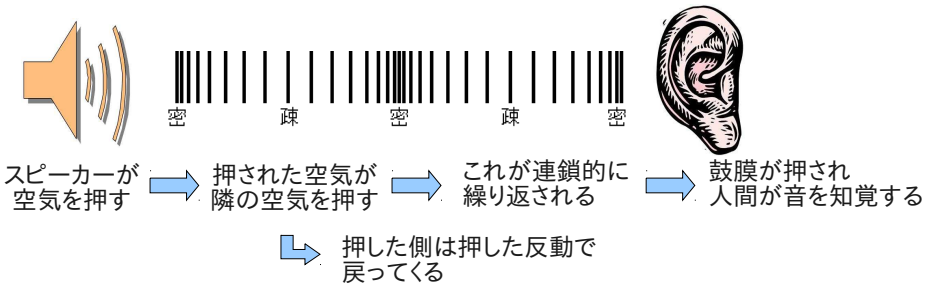
- 物理モデル合成方式
  - 楽器の発音機構をコンピュータ内でシミュレートして音を作る

デモ



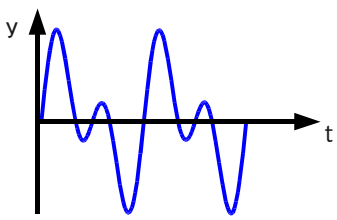
この体験授業では、このプログラムを作るための
「はじめの一歩」までに取り組みます。

そもそも音って何?



このような、ものが行ったり来たりする運動が、遥か遠くまで伝わる
現象を波動（または波）という

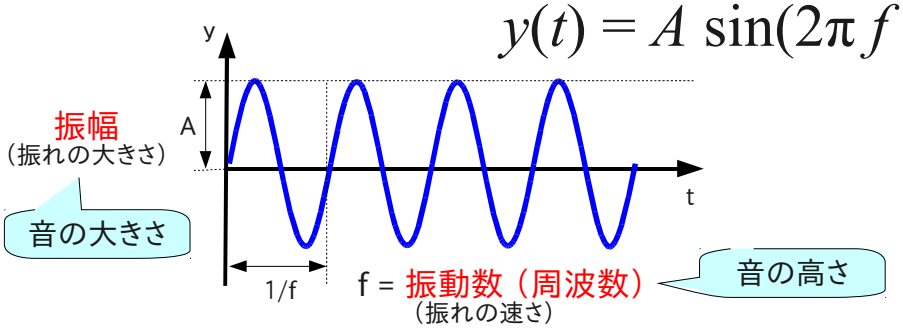
スピーカーが空気をどのぐらい
押すのかを波形として表す

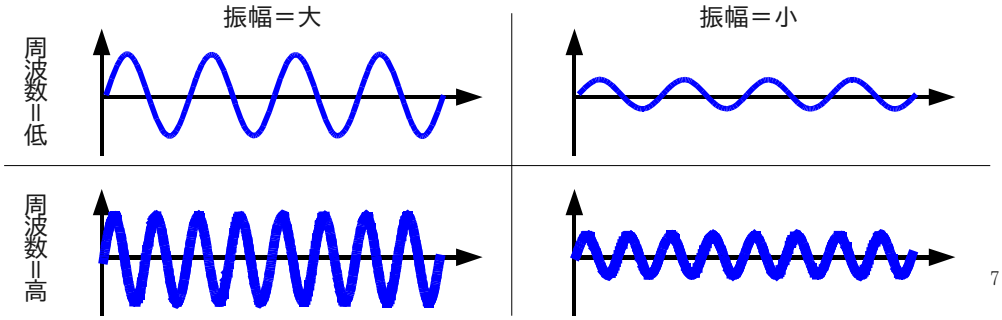


参考
- http://www.wakariyasui.sakura.ne.jp/2-2-0-0/2-2-1-1onnpa.html
- http://www.ne.jp/asahi/tokyo/nkgw/gakusyu/hadou/tate-yoko-wave/wave1.html

音の波形を数式で表してみましょう

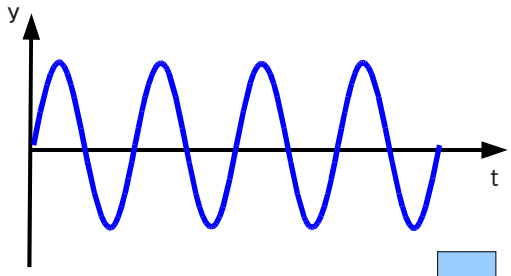
$$y(t) = A \sin(2\pi f t)$$





では、実際にこの波形を作ってみましょう

何を作ればよいのか



上の波形を、右のような
t と y の対応表として作成する

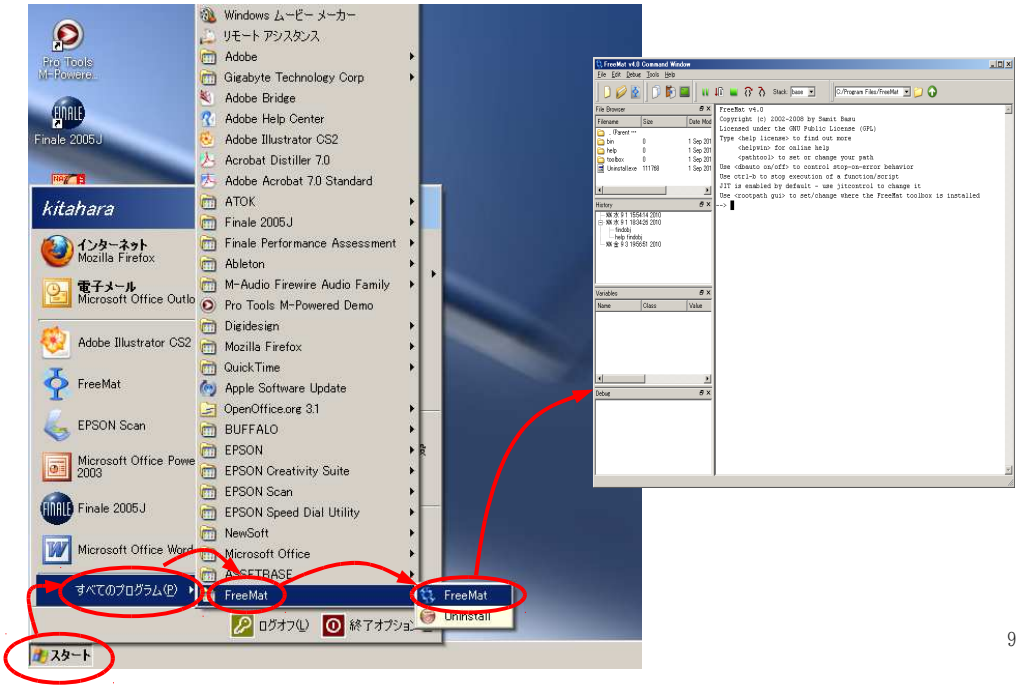
t [秒]	y
0	0.0000
1/48000	0.0057
2/48000	0.1149
・・・	・・・
10/48000	0.5446
・・・	・・・
26/48000	0.9979
27/48000	0.9999
28/48000	0.9991
・・・	・・・
54/48000	0.0314
55/48000	-0.0261
・・・	・・・
81/48000	-0.9989
82/48000	-0.9999
93/48000	-0.9979
・・・	・・・

（さらに続く）

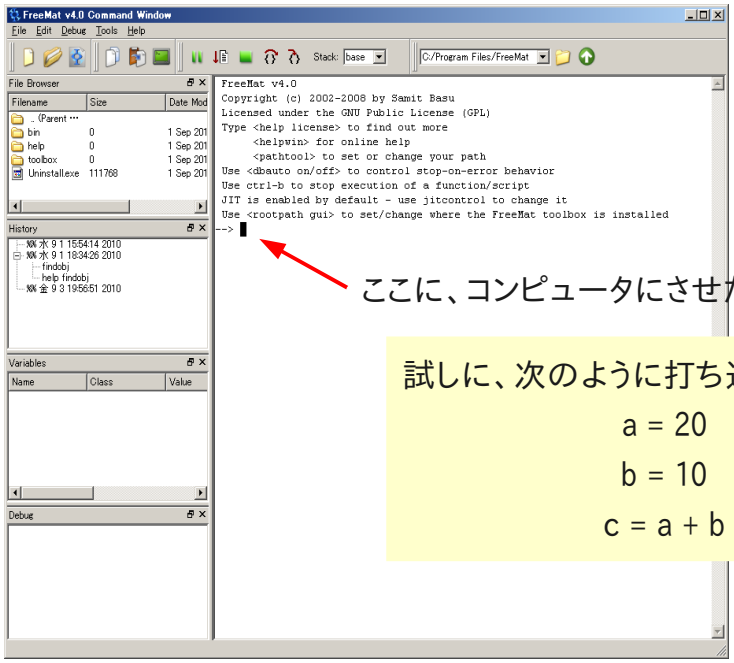
上の波形は連続波形でデータが無限にあるので、1/48000秒ごとに間引いて作成している（サンプリングという）

これをFreeMatを使って作成

「FreeMat」を起動してみよう



「FreeMat」を使ってみよう



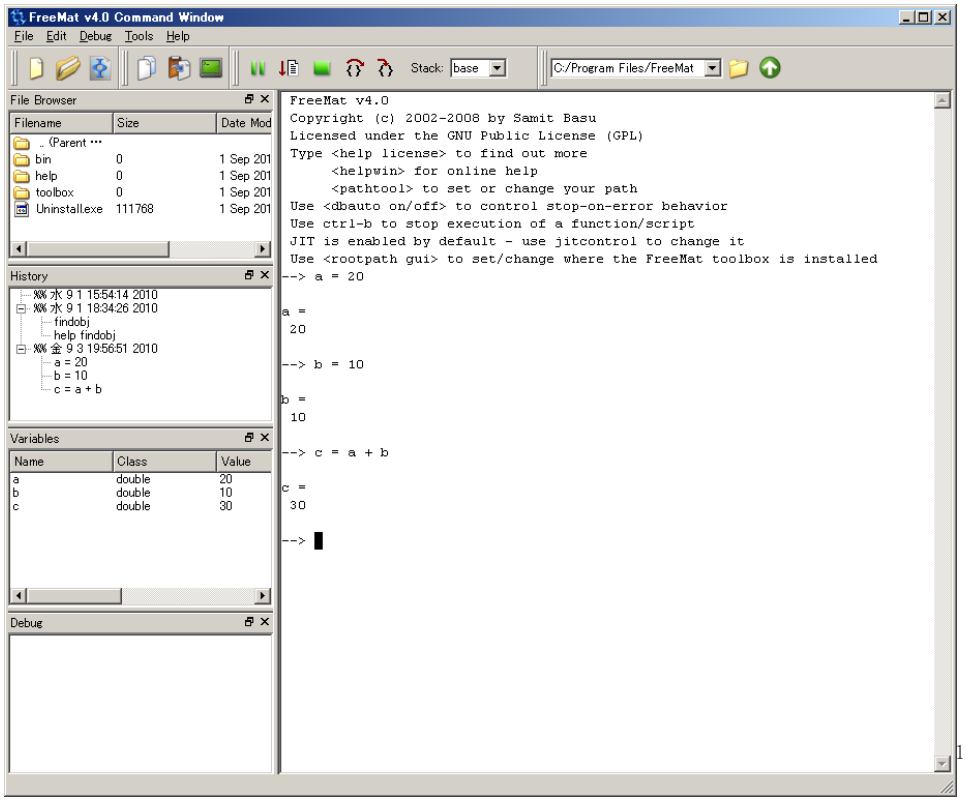
ここに、コンピュータにさせたい計算を書く

試しに、次のように打ち込んでみよう。

a = 20

b = 10

c = a + b



では、早速、音の波形を作ってみよう

FreeMatに次の式を打ち込んでみよう。

```
t = 0 : 1/48000 : 1;
f = 440;
a = 0.1;
y = a * sin(2 * pi * f * t);
```

何も表示されませんが、メモリの中にはちゃんと波形ができています。

解説

```
t = 0 : 1/48000 : 1;
```

tの値は、1/48000刻みで0から1までとします。

```
f = 440;
```

⇐ 周波数は440[Hz]とします。

```
a = 0.1;
```

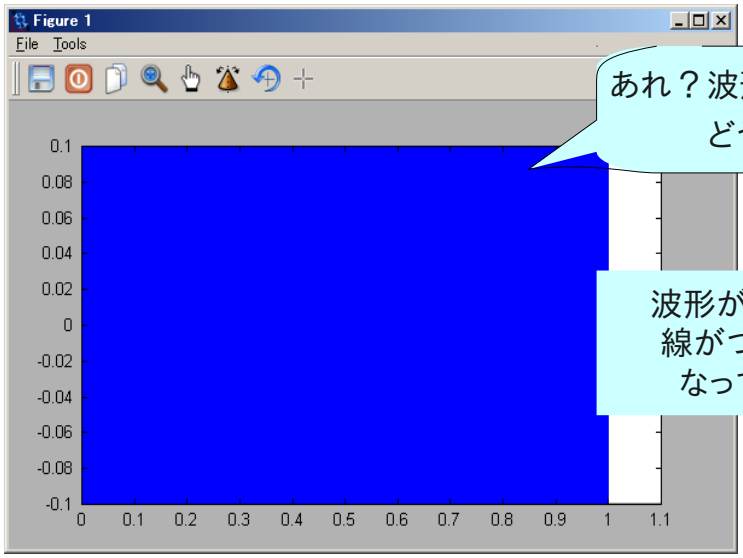
⇐ 振幅は、音が割れるぎりぎりの振幅の0.1倍とします。

```
y = a * sin(2 * pi * f * t);
```

yの値を、$y(t) = A \sin(2\pi f t)$ で計算します。

できた波形を見てみましょう

FreeMatに `plot(t, y);` と打ち込んでみましょう。



あれ？波形っぽくない！
どうして？

波形が細かすぎて、
線がつぶれて面に
なってしまった。

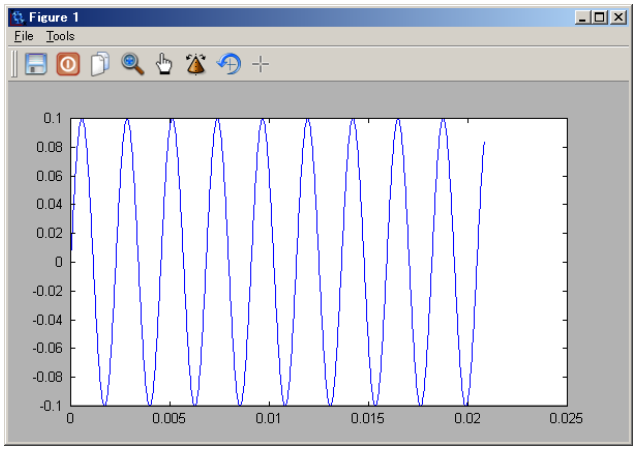
波形の一部だけ表示してみましょう

FreeMatに次のように打ち込んでみましょう。

```
plot(t(1:1000), y(1:1000));
```

「yのうち、1番めから1000番めのデータ」の意味。

「tのうち、1番めから1000番めのデータ」の意味。



できた波形を聴いてみましょう

FreeMatに次のように打ち込んでみましょう。

```
wavplay(y, 48000);
```

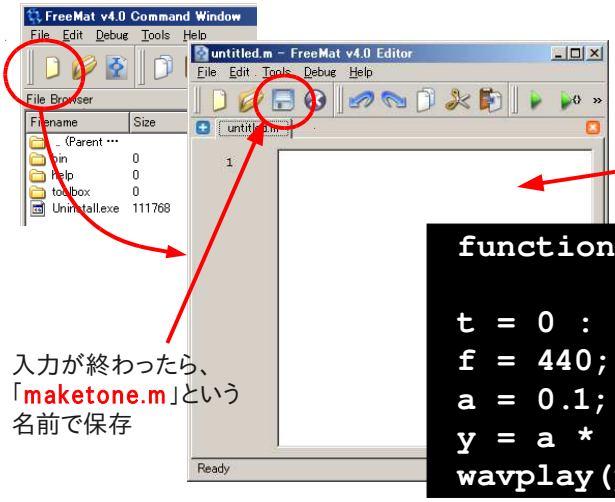
「プー」という音が、1秒間ヘッドフォンから聞こえるはずです。

注意

何も聞こえないときは、ヘッドフォンの音量設定が0になっている可能性があります。確認しましょう。

ここまでの内容をプログラムにしましょう

ここまではコンピュータにさせたい計算などを1行1行入力しましたが、毎回入力するのは面倒なので、ファイルに保存しておきましょう。
このように、コンピュータにさせたい内容を順番に書いたものを「プログラム」といいます。



入力が終わったら、「maketone.m」という名前で保存

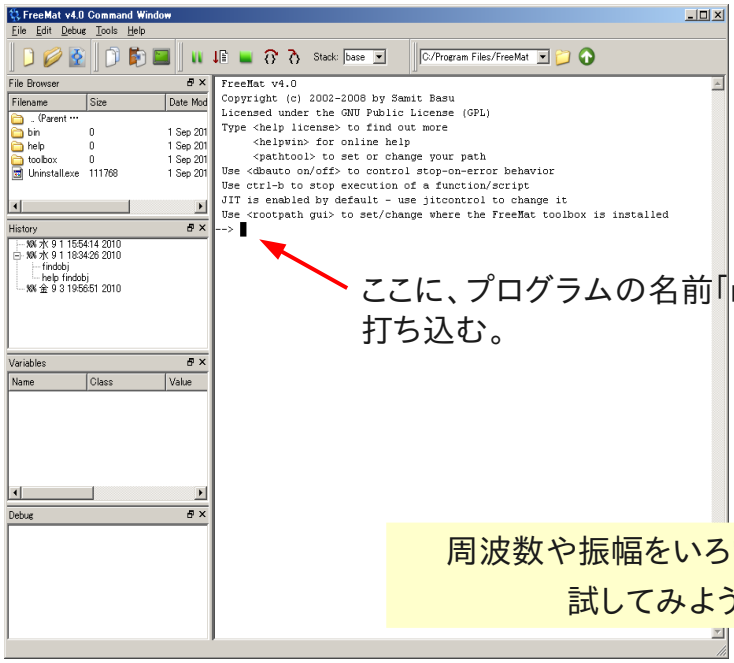
ここに以下のプログラムを入力

プログラム(1)

```
function maketone

t = 0 : 1/48000 : 1;
f = 440;
a = 0.1;
y = a * sin(2 * pi * f * t);
wavplay(y, 48000);
```

プログラムを実行してみよう



ここに、プログラムの名前「maketone」を打ち込む。

周波数や振幅をいろいろ変えて試してみよう。

2つの音を出してみよう

2つの音からなる和音を作ってみましょう。
さきほどのプログラムを以下のように書き換えて、実行してみましょう。

プログラム(2)

```
function maketone

t = 0 : 1/48000 : 1;
f1 = 440;
f2 = 660;
a1 = 0.1;
a2 = 0.1;
y = a1 * sin(2 * pi * f1 * t) + ...
    a2 * sin(2 * pi * f2 * t);
wavplay(y, 48000);
```

周波数や振幅をいろいろ変えて、
きれいな和音やきたない和音を作ってみよう。

【補足】周波数とドレミの関係

1オクターブ下へ　周波数が1/2　　周波数が倍　1オクターブ上へ

ド	130.8128	中央のド	261.6256	ド	523.2511
ド#	138.5913	ド#	277.1826	ド#	554.3653
レ	146.8324	レ	293.6648	レ	587.3295
レ#	155.5635	レ#	311.1270	レ#	622.2540
ミ	164.8138	ミ	329.6276	ミ	659.2251
ファ	174.6141	ファ	349.2282	ファ	698.4565
ファ#	184.9972	ファ#	369.9944	ファ#	739.9888
ソ	195.9977	ソ	391.9954	ソ	783.9909
ソ#	207.6523	ソ#	415.3047	ソ#	830.6094
ラ	220.0000	ラ	440.0000	ラ	880.0000
ラ#	233.0819	ラ#	466.1638	ラ#	932.3275
シ	246.9147	シ	493.8833	シ	987.7666
	[Hz]		[Hz]		[Hz]

…　…

「nオクターブ上がる」＝「周波数が2^n倍」

2つの音が1つになる!?

片方の周波数を440Hz、もう片方を880Hzにしてみましょう。

プログラム(3)

```
function maketone

t = 0 : 1/48000 : 1;
f1 = 440;
f2 = 880;
a1 = 0.1;
a2 = 0.1;
y = a1 * sin(2 * pi * f1 * t) + ...
    a2 * sin(2 * pi * f2 * t);
wavplay(y, 48000);
```

どうなりましたか?

f2がf1のちょうど2倍のときだけ、1つの音のように聞こえるはず。

f2を色々変えて、確かめてみましょう。

3つの音も1つになる!?

さきほどの音に、1320Hzの音も加えてみましょう。

プログラム(4)

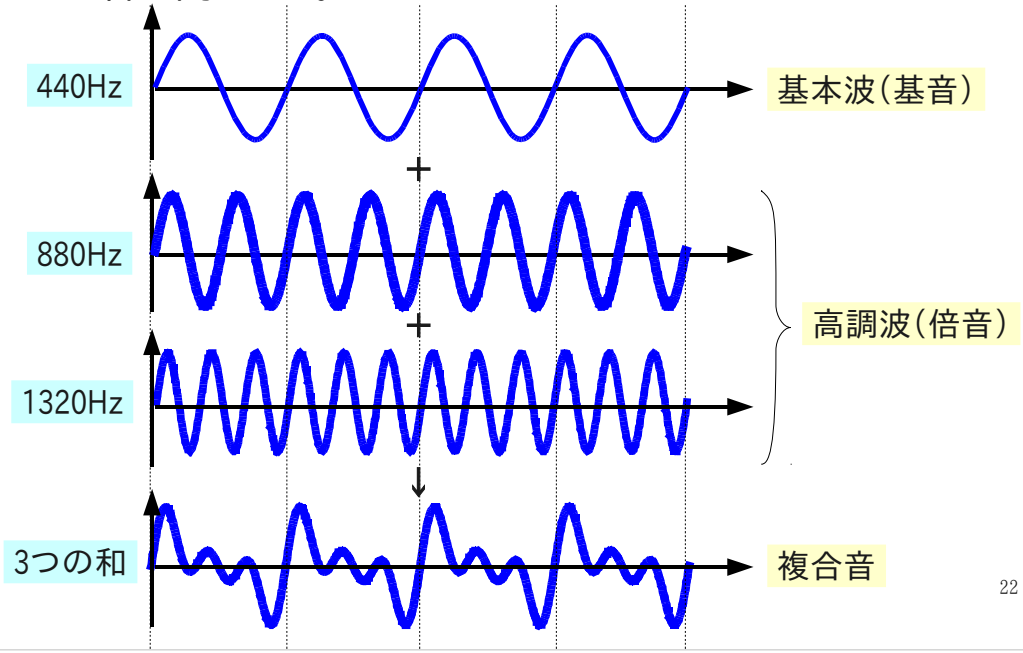
```
function maketone

t = 0 : 1/48000 : 1;
f1 = 440;
f2 = 880;
f3 = 1320;
a1 = 0.1;
a2 = 0.1;
a3 = 0.1;
y = a1 * sin(2 * pi * f1 * t) + ...
    a2 * sin(2 * pi * f2 * t) + ...
    a3 * sin(2 * pi * f3 * t);
wavplay(y, 48000);
```

どうなりましたか? こちらも1つの音に聞こえましたか?

音に関する大事な性質

ある正弦波に、周波数がその整数倍の正弦波を足し合わせると、1つの音に聞こえる。



やってみよう

1. 「プログラム(3)」の f2 を少しずつ変えてみて、どのぐらいずれたら2つの音に聞こえるようになるか、実験してみよう。
   結果

2. 「プログラム(3)」または「プログラム(4)」の振幅(a1, a2, a3)をいろいろ変えると聞こえ方(音色)がどのように変化するか、実験してみよう。
   結果

まとめ

- 整数倍の周波数の正弦波を足し合わせると、1つの音に聞こえる
- それぞれの正弦波の振幅を変えると、音色が変わる
  - 高い周波数の正弦波の振幅が大きいと、甲高い音に(トランペットのような)
  - 高い周波数の正弦波の振幅が小さいと、丸い音に(フルートのような)
- ただし、これだけで世の中の楽器の音を真似できるわけではない

今回扱わなかった話題

- 音量制御(だんだん音が小さくなるなど)
- ビブラート
- 打楽器音(「音の高さ」がない)

【付録】デモで使ったシンセもどきプログラムの実行方法

1. 右のプログラムを入力し、「mysynth.m」という名前で保存する。

```
function mysynth

global sliders

fig = figure

sliders = [ makeslider(20, 20, 's1')
            makeslider(40, 20, 's2')
            makeslider(60, 20, 's3')
            makeslider(80, 20, 's4')
            makeslider(100, 20, 's5') ]

buttons = [ makebutton('Do', 20, 180, '262')
            makebutton('Do#', 50, 140, '277')
            makebutton('Re', 80, 180, '294')
            makebutton('Re#', 110, 140, '311')
            makebutton('Mi', 140, 180, '330')
            makebutton('Fa', 200, 180, '349')
            makebutton('Fa#', 230, 140, '370')
            makebutton('So', 260, 180, '392')
            makebutton('So#', 290, 140, '415')
            makebutton('La', 320, 180, '440')
            makebutton('La#', 350, 140, '466')
            makebutton('Ti', 380, 180, '494') ]


function h = makeslider(x, y, tag)

h = uicontrol('Style', 'slider', 'Position', [x, y, 10, 100], ...
              'Tag', tag)

function h = makebutton(text, x, y, f)

h = uicontrol('Style', 'pushbutton', 'Position', [x, y, 50, 30], ...
             'String', text, 'Callback', ['maketone2('  f  ')'])
```

2. 右のプログラムを入力し、「maketone2.m」という名前で保存する。

```
function maketone2(f)

global sliders

fs = 48000;
t = 0 : 1/fs : 1;

a1 = 0.1 * get(sliders(1), 'Value');
a2 = 0.1 * get(sliders(2), 'Value');
a3 = 0.1 * get(sliders(3), 'Value');
a4 = 0.1 * get(sliders(4), 'Value');
a5 = 0.1 * get(sliders(5), 'Value');

y = a1 * sin(2 * pi * 1 * f * t) + ...
    a2 * sin(2 * pi * 2 * f * t) + ...
    a3 * sin(2 * pi * 3 * f * t) + ...
    a4 * sin(2 * pi * 4 * f * t) + ...
    a5 * sin(2 * pi * 5 * f * t);

wavplay(y, fs);
```

3. FreeMatの命令を打ち込むところに、

```
mysynth
```

と打ち込む。

mysynth.m と maketone2.m は、私のWebサイトからもダウンロードできます。