2014.11

データフローによるサウンド・プログラミング

Pure Data (Pd)入門 1

Data Flow Sound Programming ~Pure Data Primer 1

小泉 宣夫

このテキストは初めて Pure Data を学ぶ人のためのものです。サイン波による合成と波形テーブル合成まで、パッチの作り方を説明しています。

This text is prepared for beginner to learn programming in Pure Data. As a goal, fundamentals of patch programming for Sine wave additive synthesis and wave table synthesis will be learned.

１．準備　Preparation

■Pure Data とは

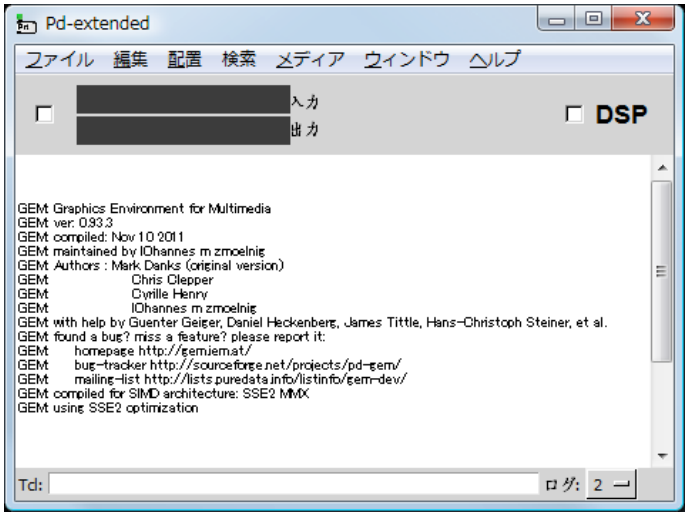
Pure Data (Pd)はデータフロー形のグラフィカルなプログラミング環境である．画面上に部品をならべ，それらを線でつないでいくことにより，データの流れ（フロー）をつくることができる．音響信号に対してはリアルタイムで出力可能な処理を行うことができる．フリーウェアとして多くの開発者に支えられて拡張されており、それらは原作者 Miller Puckette 自身によるリリース Pure Data Vanilla に対し、Pure Data Extended として配布されている。ここでは Pure Data Extended（以降 Pd と略す）を用いることにする。説明は Windows 用の Ver.0.43.4 に基づいている。Vista，7，　あるいは 8 で動作している。

■インストール方法

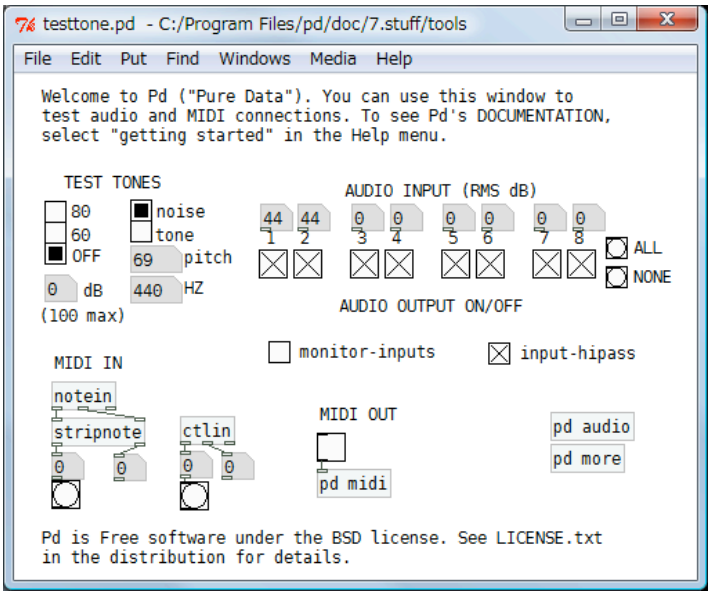
インストーラ（Pd-0.43.4-extended-windowsxp-i386 など）をダブルクリックすると自動的にインストールされる。

■オーディオの設定

Pd を起動すると以下のような画面が出る。これをコンソールという。



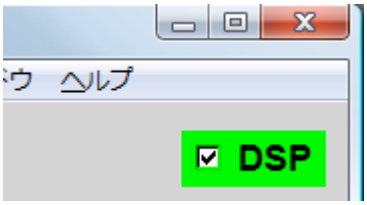
コンソールの画面



オーディオテストの画面

メディア（Media）->オーディオと MIDI のテスト（Test Audio and MIDI）

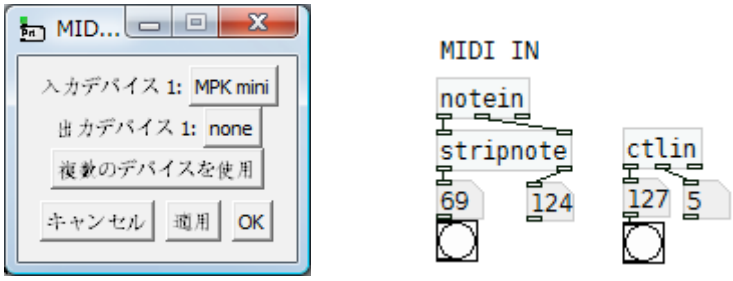
この画面は Pd のパッチの形になっている，TEST TONES のボタンを 80 か 60 にして連続的なサイン波が聞こえればＯＫである．もし，断続的に聞こえる場合はメディア（Media）->オーディオ設定(Audio Settings)を開き、遅延（delay）時間を長く設定（例えば 100ms->120ms）して連続的な音が再生できるようにする．音が聞こえない場合は出力デバイス（output device）で適切なものを選択する．



コンソール画面の DSP チェック欄

コンソール画面には DSP の表示がある。ここにチェックが入るとオーディオ信号が流れる状態になる。オーディオ信号が出ない場合はここにチェックが入っているがどうかを確認する。オーディオテストの場合は自動的にチェックが入るはずである。

■MIDI 機器の接続



MIDI コントローラを接続した場合はメディア->MIDI 設定で入力デバイスを設定する。

次にオーディオと MIDI のテストを開き、MIDI IN の部分を確認する。鍵盤をたたくと stripnote の下

の左側に MIDI ノート番号、右側にベロシティ（叩く強さ）が表示される。またつまみをまわすと ctlin の下の左側につまみの値（0～127)、右側にチャンネルナンバーが表示される。

■Pd の学び方

Help メニューのブラウザ（Browser）から Pure Data/を選択し、control examples, audio example を順番にあけて試していくと系統的に学ぶことができる（ただし、英語である)。また部品の解説としては、パッチ作成時に object（オブジェクト）を書き、それを右クリックして help をあけると解説画面がでてくる。またその解説画面自体がパッチになっている。Reference からも object の使い方を学ぶことができるが、すべて網羅されているわけではない。また reference/Gem/からは画像処理の例題をたどることができる。

オーディオ系のチュートリアルとしてはフォルダーメニューの Documentation 欄にある FLOSS Manual が短くまとまっている。Computer Music に関しては同じところから参照できる Programming Electronic Music in Pd が役に立つ。

■注意すべき点

プログラム（パッチという）を複数同時にあけると機能が衝突して動かなくなることがあるので、必要のないものは閉じる。また日本語名のフォルダに格納すると動作しないことがあるので、プログラム（パッチ）やフォルダはアルファベットで名前を付ける。パッチは一つのフォルダにまとめてドキュメントフォルダー直下に置き、その中にすべて納めておくようにする。

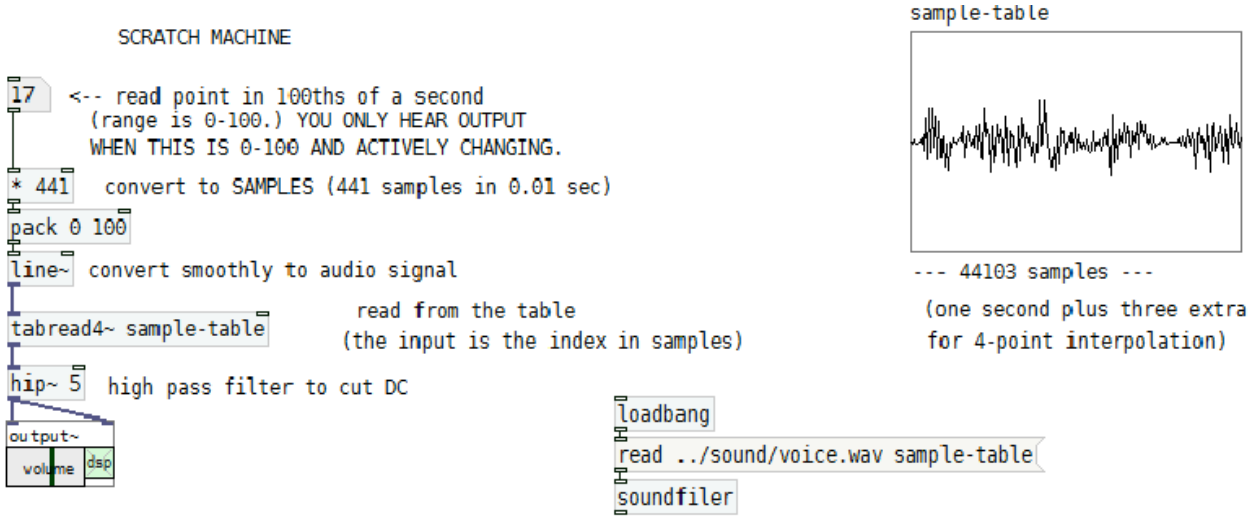
■OS の違いによる互換性

パッチはテキストで書かれており、OS が違ってもほぼ互換性があるが、拡張機能については OS によって異なるものがある。

■ヘルプブラウザーの例題から　スクラッチマシン　B07.sampler.pd

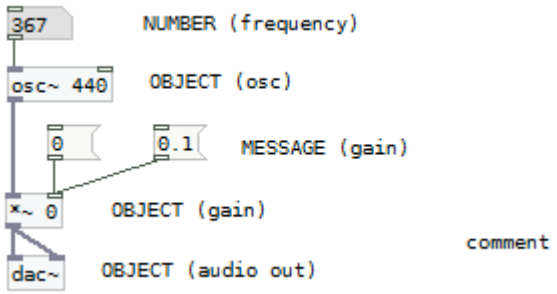
ヘルプブラウザーから Pure Data/3.audio.examples/B07.sampler.pd を開けてみよう。

これはサンプルされたデータの再生速度を変更し、スクラッチ再生することができるパッチである。左側のスライダーを動かすことで音声が再生される。



２．サイン波の生成　Sine Wave Generator

■Number, object, message, comment の使い方 Osc001.pd



では、さっそく音をだす例題から始める。ファイル（File）＞新規（new）をクリックするとプログラム作成用のウィンドウが開く。これをパッチ(patch)あるいは canvas という。ここに配置（put）メニューで部品（number, object, message, comment）をならべて、それらをつないでいく。では図にならってつくっていこう。それぞれの部品（number, object, message など）は囲みのかたちが違うので容易に識別できる。

ナンバーnumber：

数値を扱う。マウスを上下にドラッグすると数字がかわる。受け入れた数値を常に表示する。

オブジェクト object：

プログラムを構成する重要な部品で、用意されている。自分で作成したパッチを object として扱うこともできる。

メッセージ message：

数値やテキストを object に送りこむための部品。

コメント comment：

注釈をいれることができる。アルファベットが使いやすい。

-部品の接続

部品の上側の辺に並ぶものが入力 inlet、下側の辺に並ぶものが出力 outlet である。outlet と inlet を線でつないでいくことにより、プログラムが機能する。

-編集モードと実行モードの切り替え

Edit メニューの edit mode をクリックするたびに、編集モード（手のカーソル）と実行モード（矢印カーソル）が切り替わる。部品を加えたり編集したりするときは編集モードにする。実行するときは実行モードに切り替える。

-オーディオ信号動作のオン、オフ

作成したら、まずコンソールの DSP にチェックを入れてオーディオ信号が処理される状態にする。編集する時には DSP をオフにする。

-オーディオ信号とメッセージデータの違い

扱われるデータは数値や文字などによるメッセージデータと設定したサンプリング周波数（44100Hz）によって流れ続けるオーディオ信号の 2 種類がある。接続する線を見ると、オーディオ信号を扱う線は太くなっていることを確認する。またオーディオ信号を扱うオブジェクトにはかならず～（チルダ、tilde）がついていることにも注意する。

-パッチの説明

osc~：サイン波発振器

サイン波（正弦波信号、純音）を生成する object で、発振器(Oscillator)の略。引数 440 は初期設定の周波数を表すが引数はなくてもよく、Inlet から受けとる数字に置き換わる。一番上の number の上をカーソルで上下させると数字が変化し周波数を変えることができる。すべらせて 440 あたりにしておく。

＊～：信号の掛算

オーディオ信号に対して掛算を行う object である（メッセージデータの掛け算に対しては*を用いる）。0 はデフォルトで 0 の掛け算を意味しており、オーディオデータはここで 0 にされている。message は 0 と 0.1 を用意している。0.1 をクリックすると object のデフォルト値であった 0 が 0.1 に入れ替わり、0.1 倍の掛け算が実行される。すなわち絞った音量の音が出力される。0 をクリックすると 0 倍の掛け算に代わり、音はとまる。

dac~：オーディオ出力(digital to analog conversion)

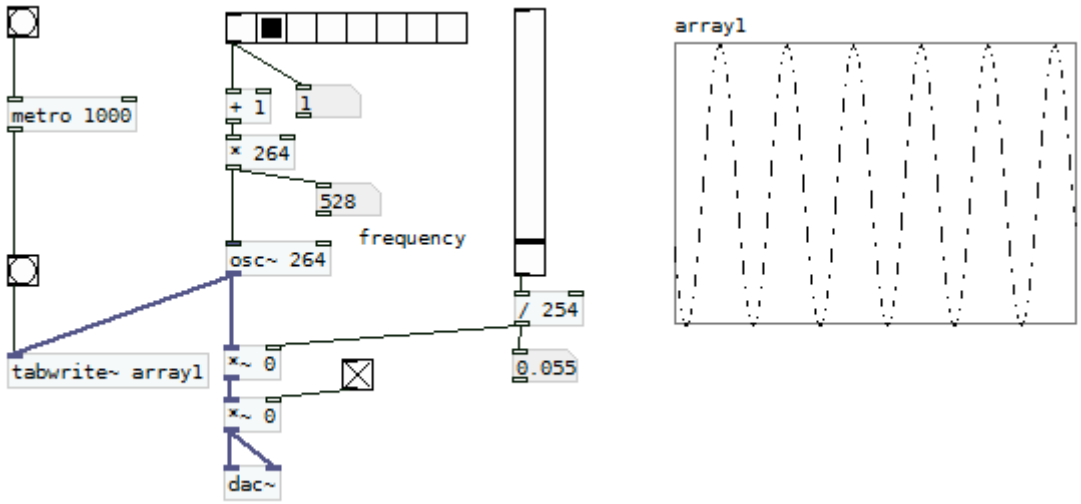
オーディオ出力。2 本の inlet は左右のチャンネルを表す。

–実行の手順

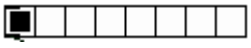
作成が終わったら、edit mode をクリックして実行モード（矢印カーソル）にする。一番上の number を 440 あたりにし、メッセージの 0.1 をクリックし、DSP にチェックを入れるとサイン波の音が聞こえる。マウスを number の上で上下にドラッグすると音の高さが変化することを確認する。

3．操作・表示部品の使い方　Using graphical elements

■ボタン、スライダー、波形の表示　　Osc002.pd



サイン波の周波数を倍数で変化できるようにして、さらにその波形を表示する。put メニューで新しい部品を用いる。

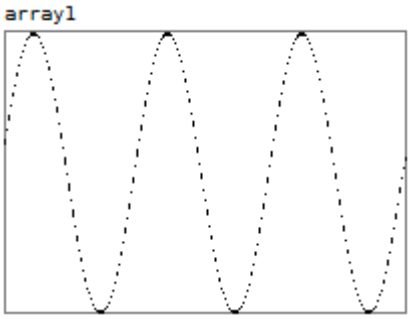
hradio　：  水平ラジオボタン

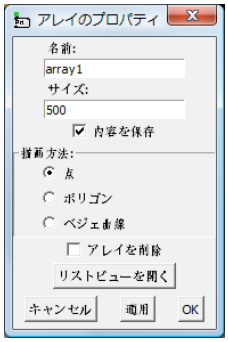
横にボタンスイッチが並んでおり，デフォルトでは左から 0 から 7 までの数字を出力する．ラジオのボタンスイッチから，この名前が付いている．vradio は垂直にボタンが並んでおり、機能は同じである。

vslider：垂直スライダー

値を変化させるためのスライダー。デフォルトでは 0 から 127 まで変化する．右クリックでプロパティを見ると変化する値を確認することができる。プロパティを編集して値の範囲をかえることもできる。hslider は水平方向のスライダーで、機能は同じ。

array：アレー





数値を表すグラフを表示する。プロパティの size を，デフォルトの 100 から 500 にする．画面の大きさはかわらないが表示されるデータ数が 100 点から 500 点になる。

bang： バング=トリガー

□の中に○が入っている部品で起動用のトリガーをかける。途中に入れることによりトリガー信号の状況を確認することもできる。

toggle：□トグル＝オン・オフ

bang と同様四角いボタンのかたちをしている。1 か 0 のメッセージ切り替えて発する。ここでは音量スライダーの後にもう一度音のオンオフ機能を加えている。ただし、コンソールの DSP スイッチとは別であり、信号の振幅を 0 に抑えているにすぎない。

-新たな object

tabwrite~ array 名：信号の書き込み

array にオーディオ信号の波形を書きこむ

metro：メトロノーム

ミリ秒の数値を指定するとその間隔でトリガー信号をだす。メトロノームの意味。

+ - * / : 加減乗除

それぞれ加算、減算、乗算、除算。オーディオ信号に用いる場合はそれぞれの後に~をつける。

このパッチはドの音の高さ 264Hz のサイン波をつくり，264Hz の倍数にある音をスイッチの切り替えで作りだせるようにしている。264Hz はド（C）の音である。

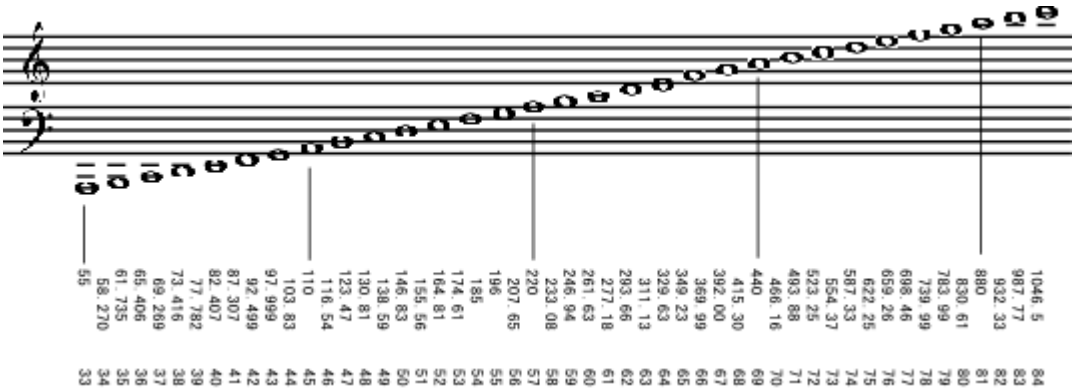
また，音の大きさを自由に変えられるようにする．put メニューで vslider を選択する。デフォルトではスライダーを動かすことにより値が 0 から 127 まで変化する．出力を 254 で割り，最大振幅が 0.5 まで音の出力が変えられるようにする．

最後に信号波形が見えるようする．put で array を選択する．プロパティの size を，デフォルトの 100 から 500 にする．array に波形を書かせるための object は”tabwrite~ array 名”である．ただし波形を繰り返し更新させる必要がある．そこで繰り返しトリガーをかける object ”metro 時間（ms）”を用いる。は, 指定した時間（ms）毎にトリガーを出すメトロノームである. メトロノームを起動するために bang(バング)を置いている．Bang はトリガーをかけるメッセージである。Metro で時間（ms）を設定すると、周期的に Bang メッセージが送られる。metro の後にも bang を入れると点滅してトリガーの様子がわかる．

ではサイン波の周波数が１，２，３，４，５，６，７，８倍と変化すると，音の高さはどのように感じられるかを確かめよう．１，２，４，８倍でオクターブの変化が生まれることを確認する．

４．MIDI ノートナンバーによる音階　MIDI Note Number Scale

■MIDI ノートナンバーによる音階　Osc002a.pd



周波数と MIDI ノートナンバー

長音階が演奏できるようにしてみよう。

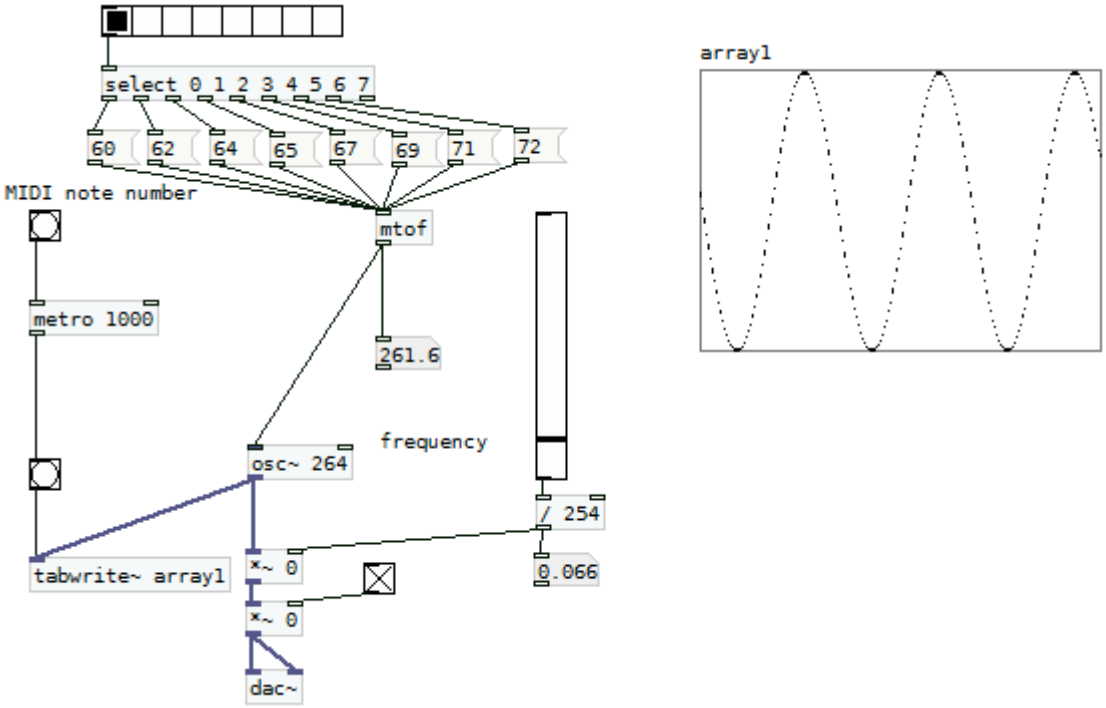
Mtof:MIDI ノートナンバーを周波数に変換（MIDI to frequency）

MIDI は音楽情報を扱う規格である。音高は半音間隔で番号が割り振られていて、MIDI ノートナンバーという。440Hz（A）は 69、1 オクターブで 12 増減する。指数を使えば、数式で周波数に変換することができるが、オブジェクト mtof（MIDI to frequency）が用意されている。

select(sel)：

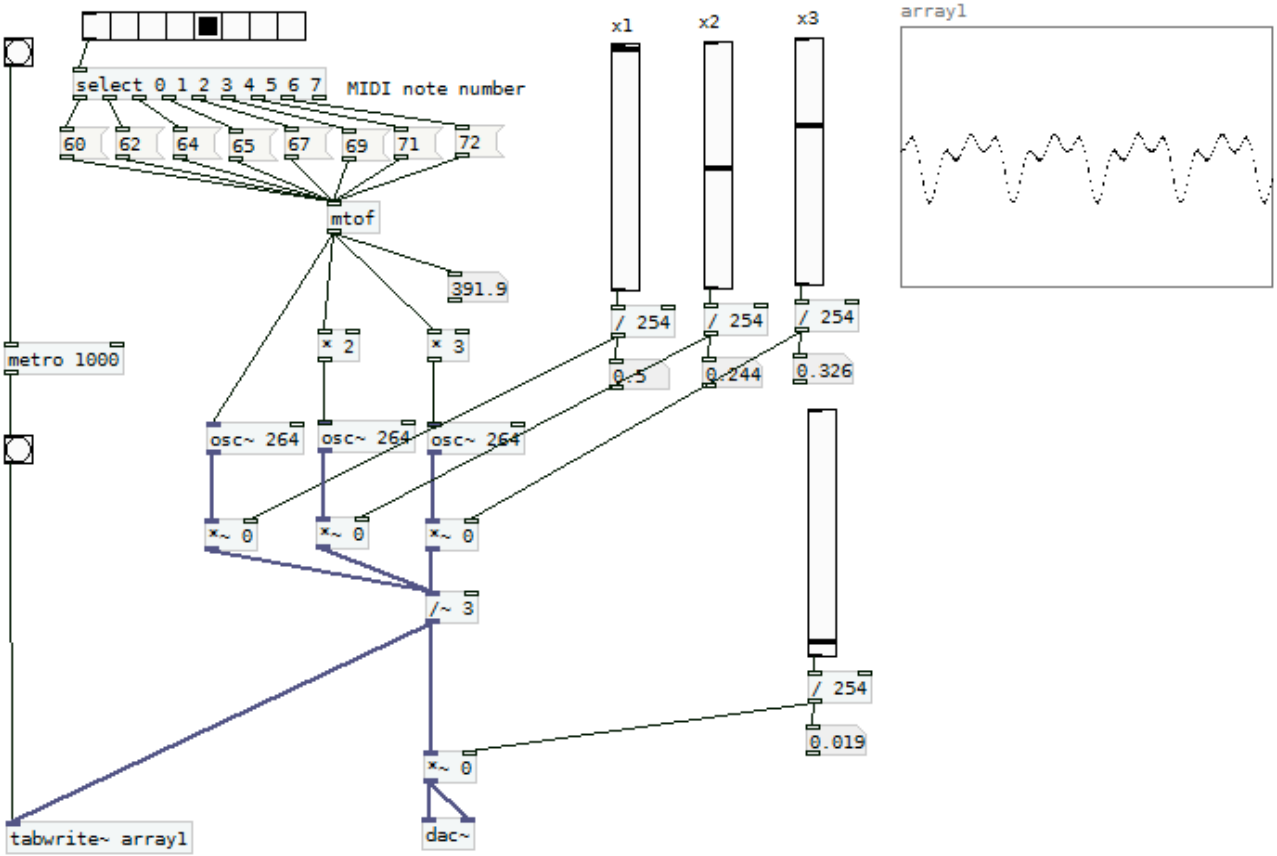
inlet の値と一致した数値の順番に相当する outlet から bang を送る。数値に合致しない場合は一番右の outlet から値を出力する。ぶらさがった message には長音階に相当する MIDI ノートナンバーを用意している。

mtof（MIDI to frequency）：MIDI ノートナンバーを周波数に変換する。



５．サイン波の倍音加算　Adding Sine Wave Harmonics

■加算合成（倍音成分の調整による音づくり）　Osc002d.pd



倍音成分を調整するパッチ Osc002d.pd

ある周波数とその２倍、その３倍の周波数のサイン波を任意の大きさで合成する。前の例に２倍と３倍の周波数の osc~を追加し、全体のボリューム調整をつける。

すべての周期的な信号はそれと同じ周期（周波数）のサイン波とその倍数の周波数のサイン波の重ね合わせとして表現できる。つまり、周期的な信号はサイン波の倍音成分を重ねてつくることができる。このようにして音を合成する方式を加算合成という。

例題では３倍までの成分を扱うが６倍まで扱えるようにしてみよう。

６．波形テーブルを用いた音の合成　Wave Table Synthesizer

■波形テーブル合成　Synth001_wt.pd

Pd で用意されている周期信号はサイン波 osc~とのこぎり波 phasor~（フェーザー）である。任意の周期信号を作る方法として考えられるのは。1 周期分の波形をテーブルに格納し、これを繰り返し呼び出して再生することである。これを波形テーブル合成という。pd では１周期分の波形は array に格納し、phasor~を使って読みだす。

-波形の格納

array を置き、名前を wavetable に、サイズを 2051 に設定する。loadbang はプログラムをロードした時点でトリガーをかける object。続く message で wavetable sinesum を送る。sinesum は倍音（整数倍の周波数による成分）をつくるもので、引数としてデータサイズ、基本周波数、そのあと順に倍数の成分の振幅値を並べることでサイン波の合成波形をつくる。ここでは 2051, 1 0.5, 0.33, 0.25 と並べることで 2051 個のデータに 4 倍周波数の成分まで使ってのこぎり波に近い波形をつくっている。このテーブルサイズであるが、補間という処理が必要なため、(2 のべき乗）+3 のおおきさを選ぶ。131, 259, 515, 1027, 2051 などである。

このメッセージを実行モードでクリックすると wavetable に波形が格納されるが、波形の最大振幅は１を超えてしまい、再生時にひずみをおこす。そこで wavetable normalize のメッセージを送り、最大振幅が 1 以内に収まるようにする。

-波形の読み出し

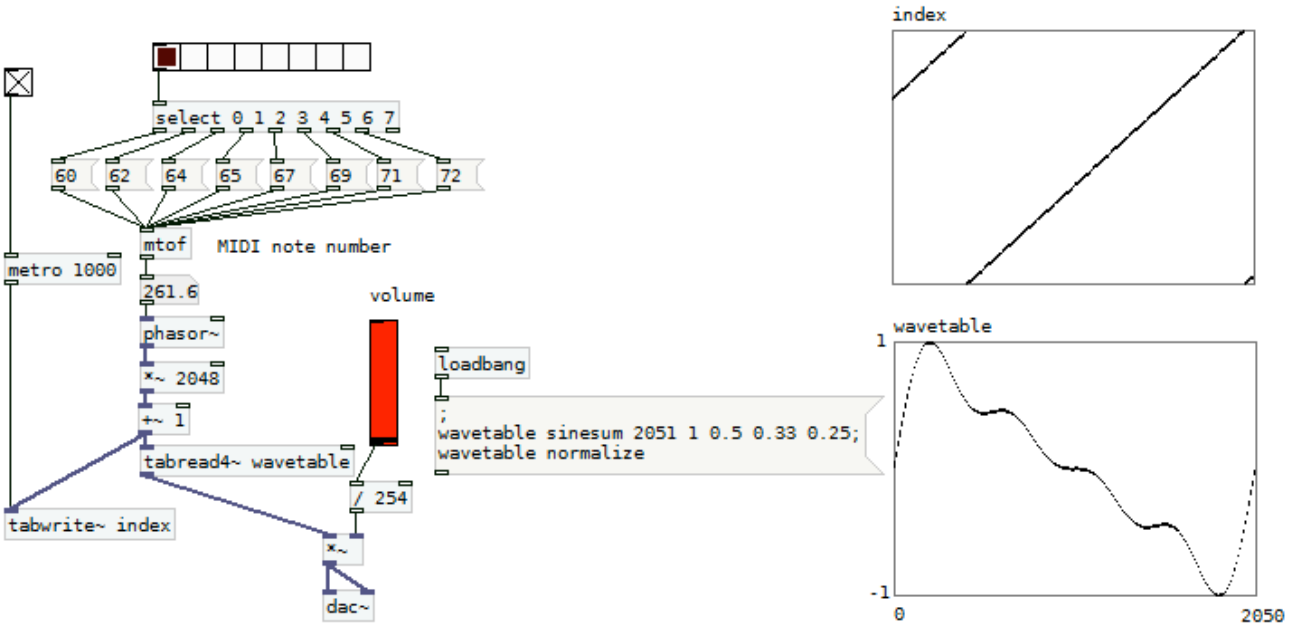
このデータを参照するオブジェクトは tabread4~である。この tabread4~は array1 の値を読み取るものである。そのために phasor~を用いる。phasor~は 0 から 1 までの変化を指定した周波数で繰り返す、のこぎり波である。この出力を 2048 倍して 1 を加えることで１から 2049 の間を繰り返す。これによって wavetable の横座標を順に参照し、すべて波形値を読み取り、これを繰り返す。サンプリング周波数の関係で読み取り点は必ずしも整数にはならないので、tabread4~は補間値が計算される関数になっており、読み取り点数より 3 つ多い個数を必要としている。この phasor~が創り出す信号を参照信号といい、index と名付けた array（プロパティで y の範囲を 2048 までにする）を用意する。

-テーブル次数の要点（補間のため）

波形を格納するテーブルの大きさは 2^n+3（たとえば 2051）、値の範囲は 0 ～ 2^n+2　（たとえば 0～2050）

phasor の振幅は 2^n（たとえば 2048）、値の範囲は 1 ～ 2^n+1　（たとえば 1～2049）

wavetable の上にマウスカーソルを置いてずらすと任意の波形が描けるので、さまざまな波形の音を聞くことができる。sinesum, normalize メッセージをクリックすれば、元の波形が復元される。



波形テーブル合成　Synth001_wt.pd

7．サブパッチ、アブストラクション　Sub Patch and Abstraction

■パッチの階層化

機能を加えるごとにパッチは複雑し、見づらくなる。パッチを見やすくしたり、パッチを再利用したりするためには、パッチを部品に分割していくことができる。それには subpatch（サブパッチ）と Abstraction（アブストラクション）という方法がある。

■サブパッチ

サブパッチは object 作成時に pd<空白> ###として###に適当名前を入れる。そうすると新たなキャンバスが開く。入力として inlet オブジェクト、出力として outlet オブジェクトを並べて作成すると、左から並べた順に inlet と outlet が pd ###オブジェクトの上下に現れるのでデータや信号はそこで受け渡しされる。音響信号を扱う場合は inlet~、outlet~を用いる。

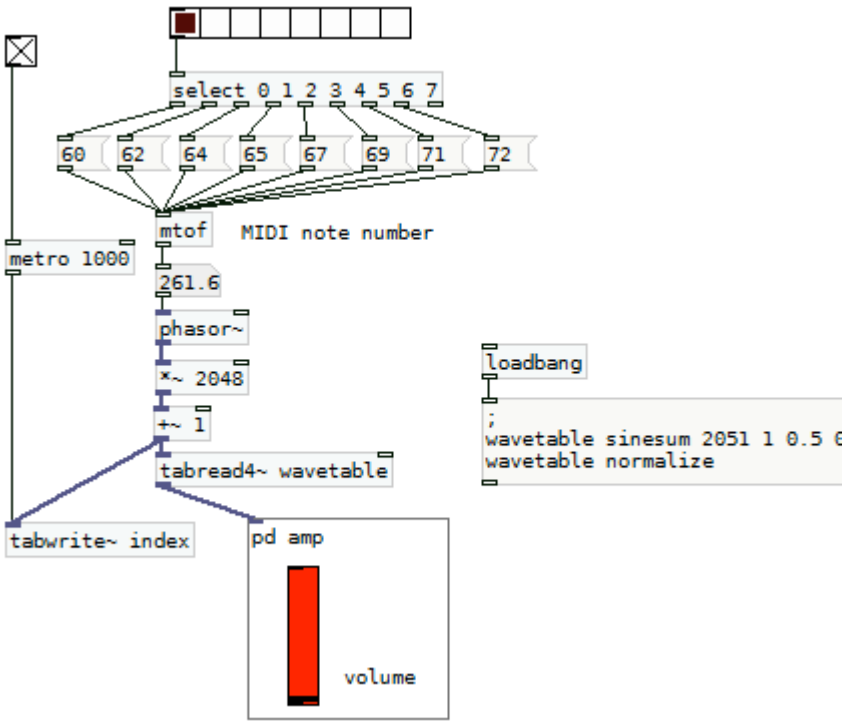
サブパッチはそのパッチの中だけで使われ、ファイルとしては表に現れない。したがってファイルの管理は複雑にならないが、流用の際は毎回コピーする必要がある。

一度つくってからサブパッチに分ける場合は pd ###と入力し、新規窓が開いたら移したい部分をカット＆ペーストする。

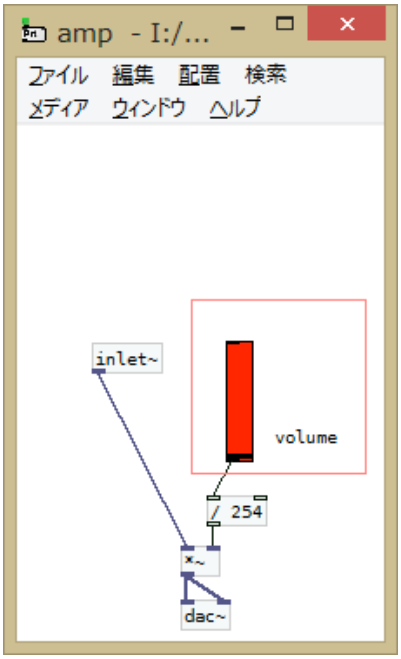
ここでは前に例題で音量コントロールの部分はサブパッチ化（pd amp）する。

■サブパッチの一部を親パッチに表示

このようにサブパッチ化する場合、スライダーなどの操作系は親パッチに表示する必要がある。そこでサブパッチのプロパティを開けて、“親パッチに表示”をチェックし、サイズを整えたら表示させたい部分を赤い枠の中に収める。この例題ではスライダーを枠内にいれることで親パッチから表示される。



Synth001_wt_a.pd 親パッチの一部をサブパッチ化する



サブパッチ pd amp

■アブストラクション

作成したパッチはそのままオブジェクトとなる。これがアブストラクションである。

パッチ内に inlet, inlet~, outlet, outlet~を置いて.pd ファイルとして保存する。次に新たなパッチを作成し、オブジェクトとして保存ファイルを参照する。そうすると inlet と outlet を通じて接続することができる。

サブパッチは親パッチと一緒に管理されて外からは見えないので、再利用の際はその都度コピーする必要がある。それに対し、アブストラクションは自前のオブジェクトとしてそのまま使える利点がある。

■結線を隠す　Hiding patch cords

サブパッチ、アブストラクションの活用でパッチを整理することができた。残る問題は結線のわずらわしさである。send(s),receive(r), send~(s~),receive~(r~),を用いると send から receive にデータが渡されるので、結線をかくすことができる。