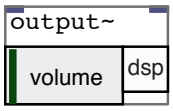
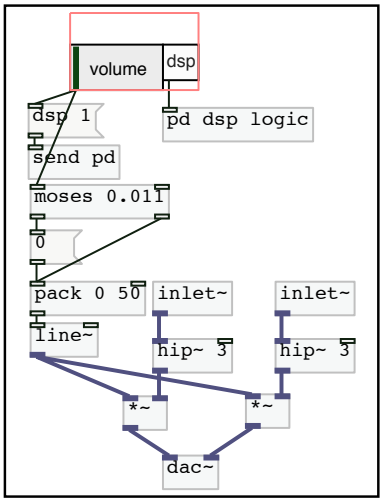
親の中身は…?

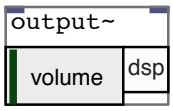
- サブパッチ化することで制御の目的のための一部のみを示すことができます
- 例えば、[output~]にはToggleボタンと水平スライダーがあります(右記上部を参照)
- [output~]を開いた(右クリックした)とき、右記の要素が赤く示した箱の中身であることを確認することができます
- これはオブジェクトをカスタマイズしたいときに有効な方法です

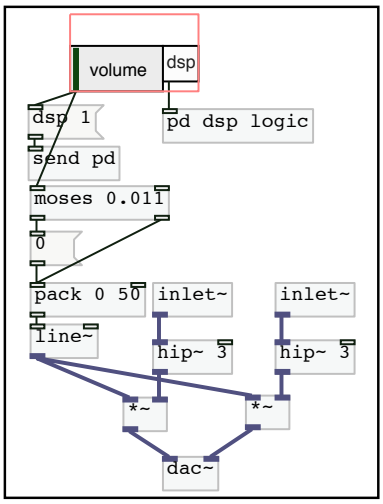




親の中身は…?

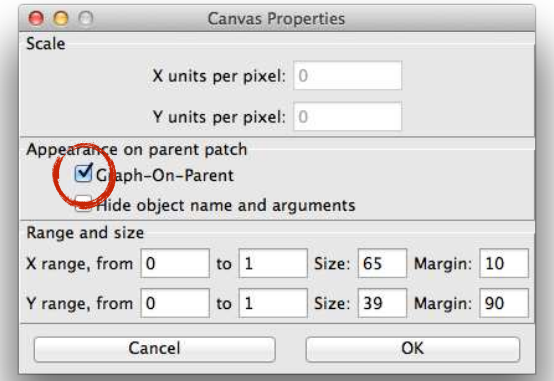
- **[moses]** は、入力された数字が制御の値(ここでは0.011)より小さい場合は左の出力端子から出力し、同等かそれ以上の場合は右の出力端子から出力します
- 引数によって基準値を設定しますが、右の入力端子へ数値を入力して変更することもできます
- [Line~] はメッセージによって定義されたレベルとタイミングを元に直線的な傾斜をつくります
- 入力されたメッセージがただ1つの場合、瞬時に上昇し、複数ののメッセージがある場合は指定された時間をかけて徐々に上昇します





親の中身は…?

- 現在のパッチを右クリックし、プロパティーを選択します
- “Graph-on-Parent.”のチェックボックスにチェックを入れてください
- ピクセル単位でサイズ設定ができます
- それぞれのマージ(余白)はウィンドウの左上を起点としています



Expr, expr~, fexpr~

- Expr群はC言語のような構文を処理するオブジェクトで成り立っています
- 複数入力の表現は; "セミコロン" で分けることで対応でき、この結果は複数出力になります
- 変数はC言語と同様にサポートされています
- 変数は実行前にvalueオブジェクトで定義されている必要があります

Expr, Expr~, Fexpr~

- Expr: データの制御
  - $i1 - $i9: 最初の9入力を整数型としてとります
  - $f1 - $f9: 最初の9入力を倍精度浮動性小数点型としてとります
  - $s1 - $s9: 最初の9入力をシンボルとしてとります (この際、シンボルはテーブルの探索に利用されます)
- Expr~: 信号制御
  - $v1 - $v9: 最初の9入力を信号としてとります (ベクトル)
- Fexpr~: サンプルレベルのデータ (例: フィルターの設計)　※当講座では扱いませ
- 説明書(保存版): http://yadegari.org/expr.html

Exprで利用可能な演算子

~	One's complement
*	Multiply
/	Divide
%	Modulo
+	Add
-	Substract
<<	Shift Left
>>	Shift Right
<	Less than (boolean)
<=	Less than or equal
>	Greater than
>=	Greater than or Equal (boolean)
!=	Not equal (boolean)
&	Bitwise And
^	Exclusive Or
|	Bitwise Or
&	Logical And
||	Logical Or (boolean)

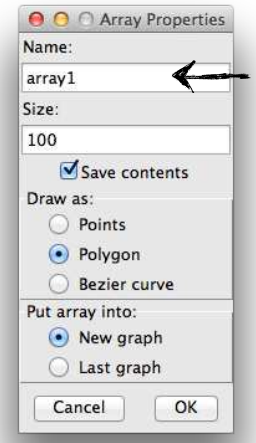
優先順位は
上から下になっています

Expr 関数

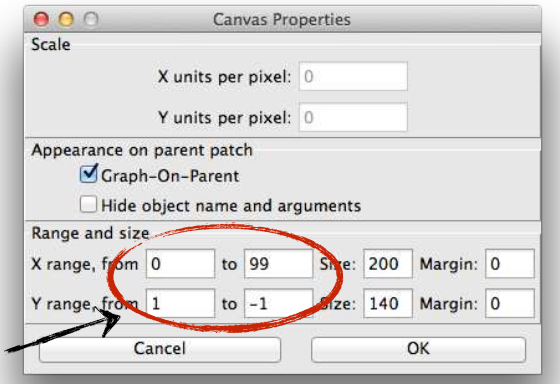
if()	3	conditional - if (condition, true-expr, false-expr)
int (), float ()	1	convert to integer/ float
rint ()	1	round a float to a nearby integer
min (), max ()	2	minumum, maximum
abs()	1	absolute value
isinf(), finite()	1	is the value infinite, finite
isnan	1	is the value non a number
copysign()	1	copy sign of a number
imodf	1	get signed integer value from floating point number
modf	1	get signed fractional value from floating-point number
drem	2	floating-point remainder function
pow ()	2	raise to the power of {e.g., pow(x,y) is x to the power of y}
sqrt ()	1	square root
exp()	1	e raised to the power of the argument {e.g., exp(5.2) is e raised to the power of 5.2}

ln() and log(), log10()	1	natural log, log base 10
fact()	1	factorial
erf()	1	error function
erfc()	1	complementary error function
cbrt()	1	cube root
expm1()	1	exponential minus 1
log1p()	1	logarithm of 1 plus
ldexp()	1	multiply floating-point number by integer power of 2
sin(), cos(), tan(), asin(), acos(), atan()	1	
atan2()	2	arc tangent of 2 variables
floor()	1	largest integer value not greater than argument
ceil()	1	smallest integer value not less than argument
fmod()	1	floating-point remainder function

配列について



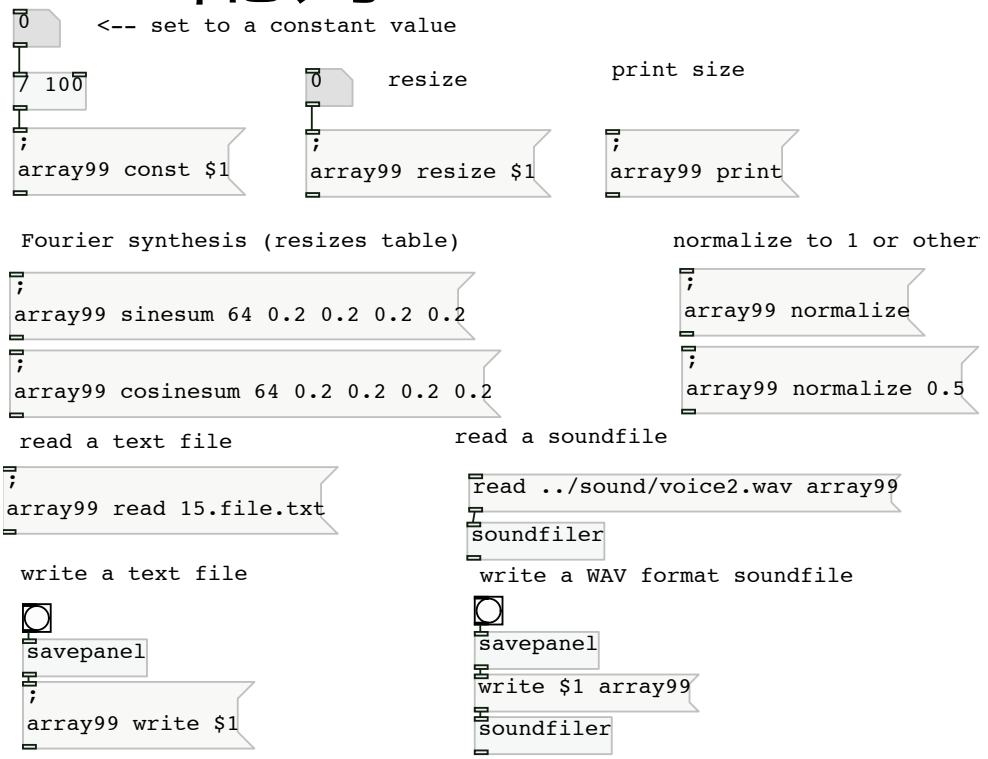
配列名の設定



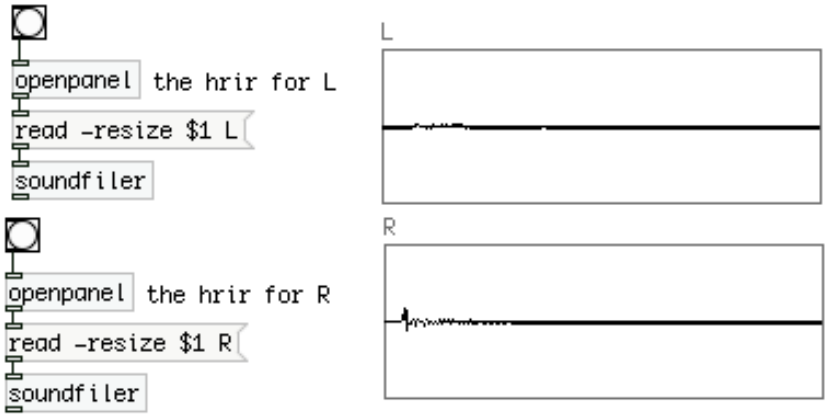
範囲の設定

- Put > Arrayで配列を作ることができます
- Arrayを利用することで同様に、数字のリストや音を扱うことができます

配列について



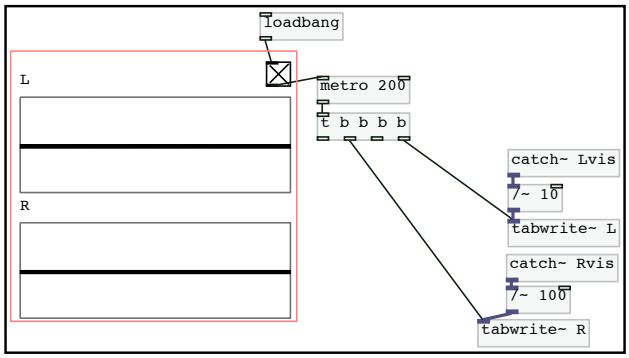
ファイルを配列に書き込むには



- **[openpanel]** はファイルを選択するためのウィンドウを表示します
- **[soundfiler]** は**[read f a(** によって、定義された配列aに格納された音声ファイルfを読み込みます
- **-resize** によって、入力したファイルと同サイズの配列をつくることができます(例を参照)

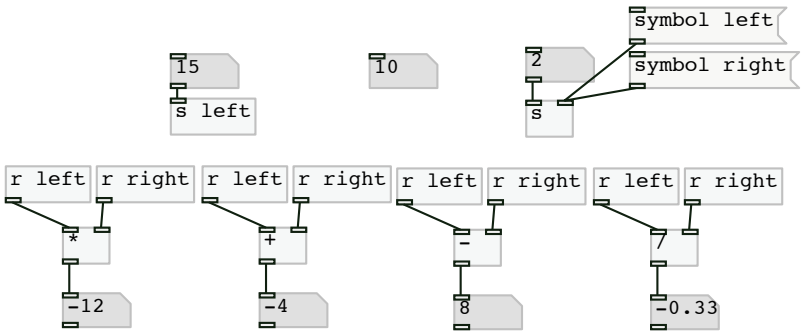
アレイの中で波形を視覚化する方法

- **[tabwrite~]** を利用して音信号を配列の中に書き込みます
- [tabwrite~]はbang[o]を入力することで、配列の最初から最後まで書き出し、表示します
- **[start X(** と **[stop(** は配列への書き込みをリアルタイムに制御します



*Visualizer.pd*の中身

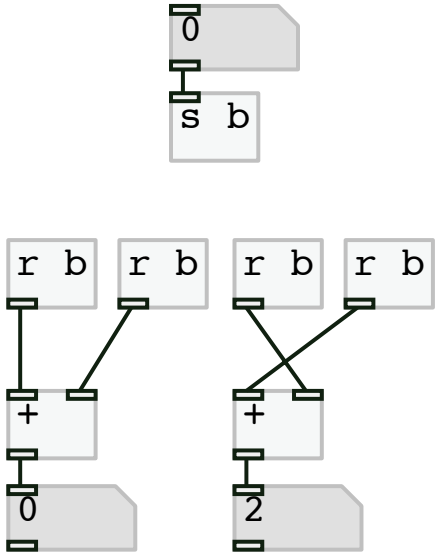
Sendの[s]とreceiveの[r]



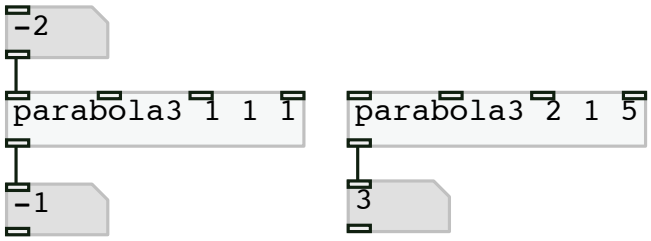
- これらのオブジェクトは接続の線を使わずに、それぞれのボックスへ接続するときによく利用されます
- これらはパッチをまたいで接続することができます
- いくつかのオブジェクトは、sendとreceiveの情報をプロパティーから設定できます
- [s]と[r]の名前は同じでなければなりません (例: [s foo] [r foo])

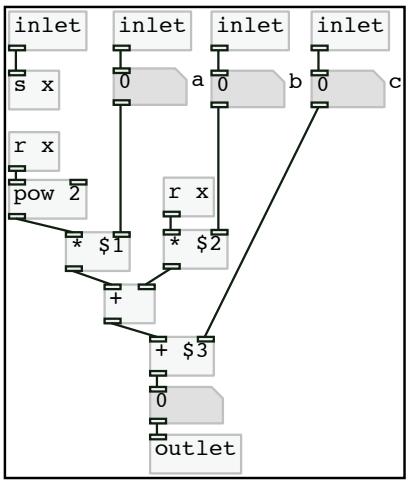
[s]と[r]の問題点

- [s]と[r]を利用することで可読性を高めることができます
- 注意して使用しなければ、実行エラーの原因となります
- では、この実行結果は何でしょうか？



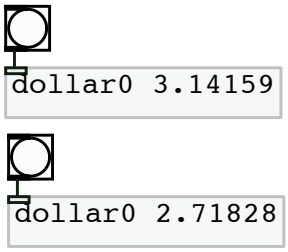
[s]と[r]の共通の参照範囲

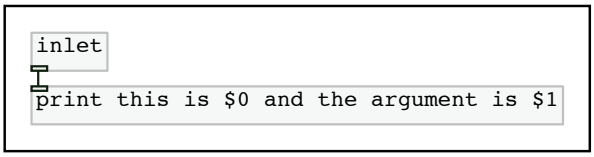




- 右のパッチには何も接続されていませんが、出力があります
- これは[s]と[r]にはパッチ内部に共通の参照範囲(スコープ)が存在するためです

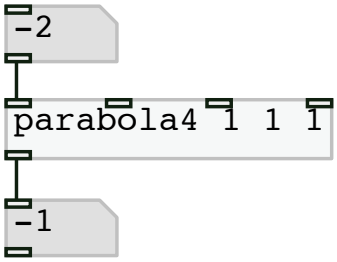
$0のおさらい

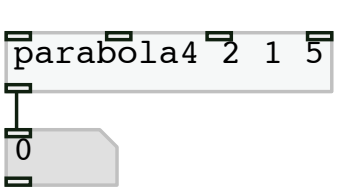


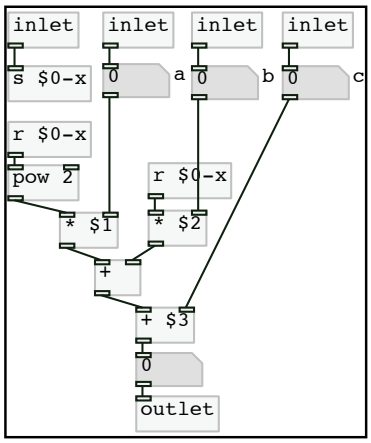


- $0はパッチまたは抽象インスタンスを、パッチまたは抽象インスタンス特有の4桁の数字で内部で、明示的に表現できます
- $0の利用によって[s]と[r]のスコープに起因する問題を解決することができます

スコープに関する問題の解決方法

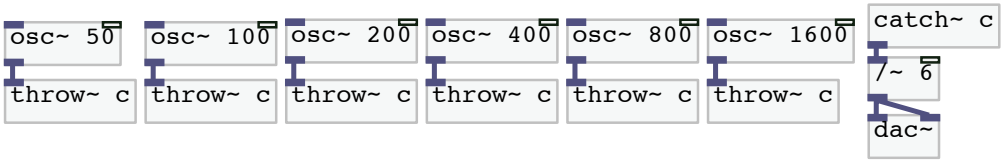






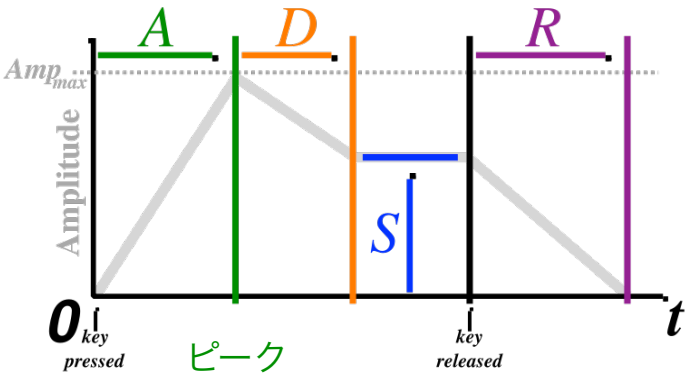
- それぞれの抽象インスタンスに[s]と[r]の名前の接頭文字として'$0-'を追加することによって、この問題を回避できます

Throw~とcatch~について



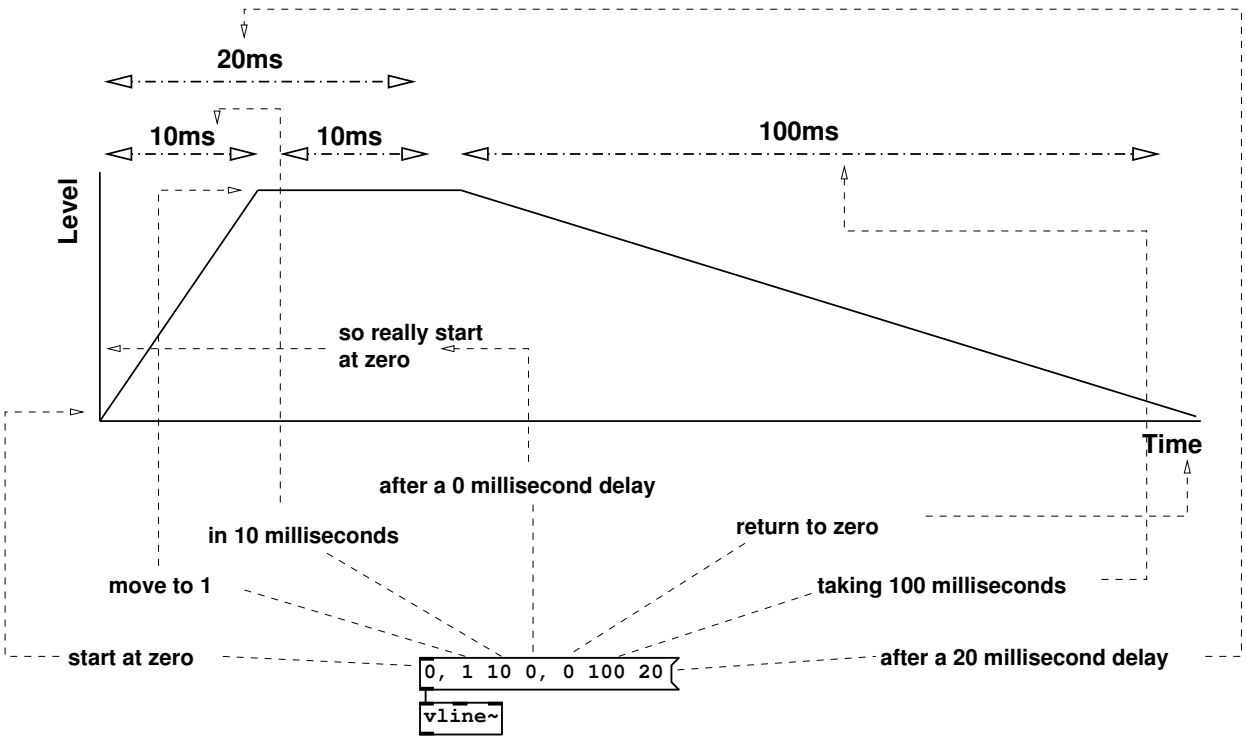
- これは音に対して利用しますが、働きはsendとreciveオブジェクトと似ています
- [throw~]オブジェクトは同じ名前をつけることができ、その場合は結合されます
- [catch~]オブジェクトはただ一つの名前である必要があります
- [throw~] は'set'メッセージによってリセットされ、再び入力を開始します

時間のエンベロープ

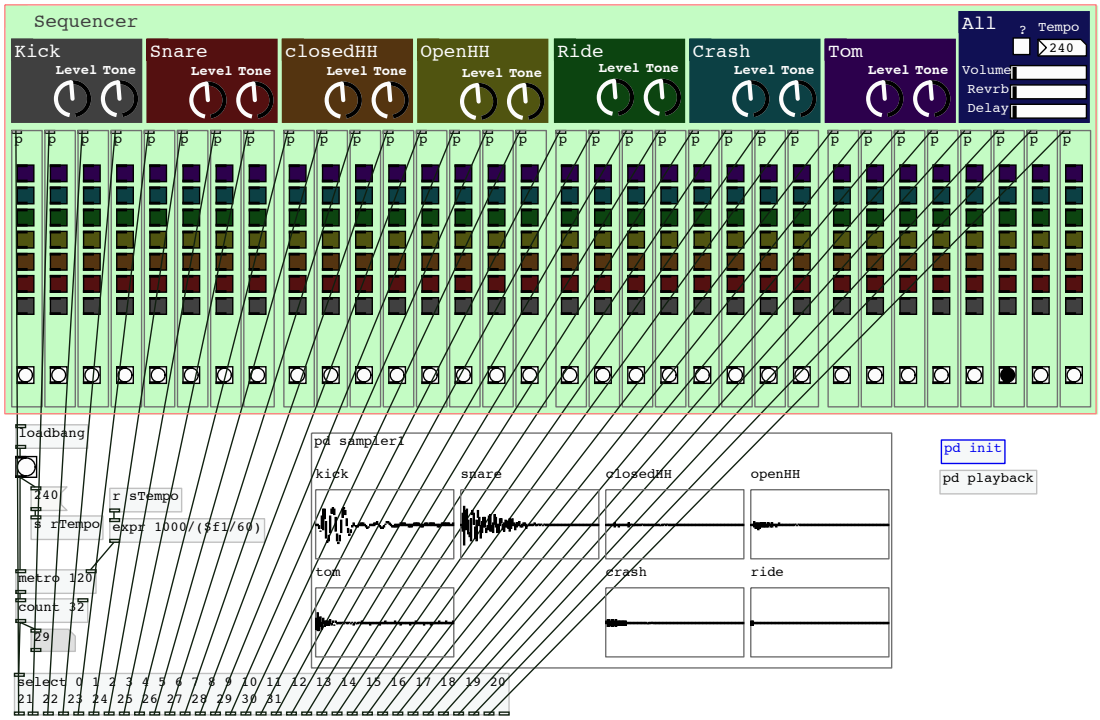


- アタック: 振幅が0からピークまでの時間
- 減衰(Decay): アタックレベルから持続レベルまでの時間
- 持続(Sustain): キーがリリースされるまでの音の持続
- リリース(Release): 持続レベルからゼロまでの時間

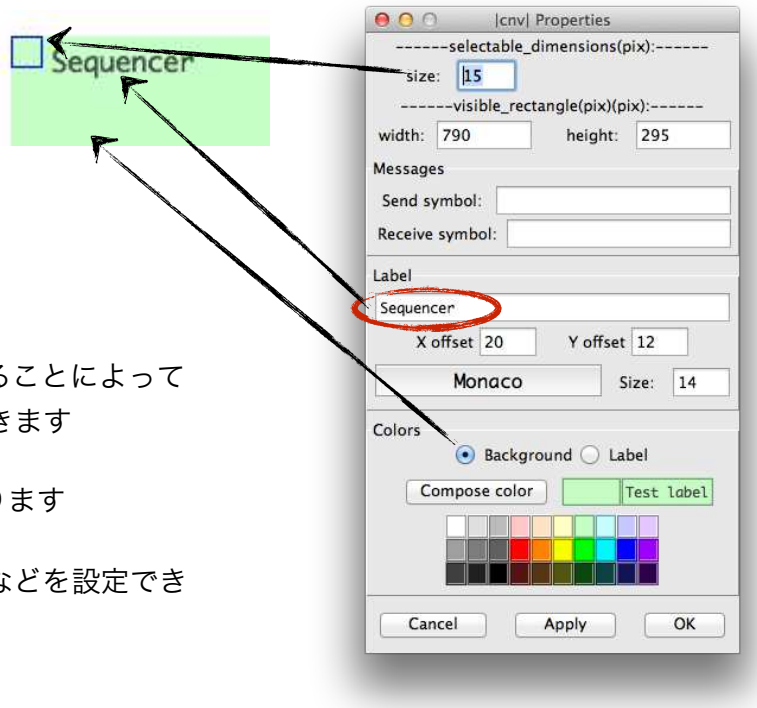
Pdにおける時間のエンベロープについて



シークエンサー

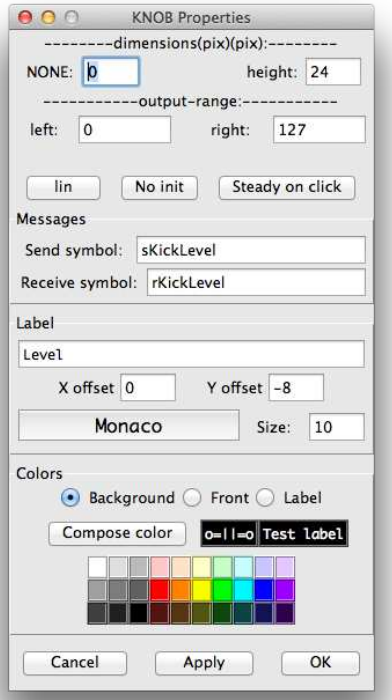


キャンバス



- キャンバスを利用することによって視覚的に見やすくできます
- GUIを作りやすくなります
- サイズやラベル、色などを設定できます

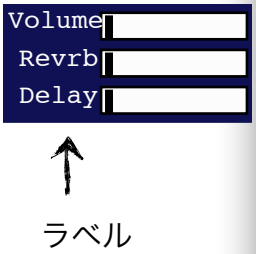
つまみ(ノブ)

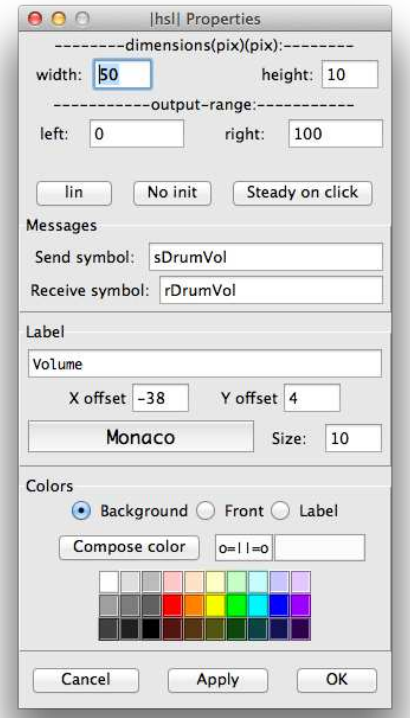


- **[knob]** は視覚的なつまみを作り、マウスで操作できるようにします
- 設定した範囲の値が出力されます
- [Knob]は値を送受信できます:
  - ここでは、送信元をsKickLevel、受信元をrKickLevelという名前にしています
  - もし両方に同じ名前を使用した場合、無限ループが始まってしまいます

スライダー

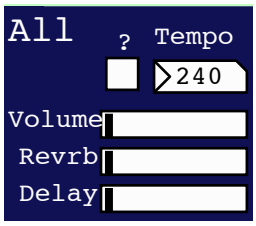
- 垂直・水平スライダーは数値を設定するのに便利です
- [knob]のように出力範囲を設定することができます
- ラベルの位置は初期状態の位置が基準です
- この例では、'Volume'ラベルを初期位置より左側へ-38 ピクセル移動したところにセットしています

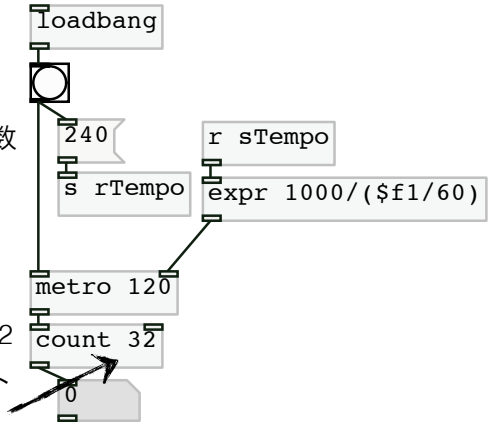




テンポの設定

- テンポは1分間あたりのビート数で表現されます(bpm)
- **[metro]** はms間隔でリズムを刻むメトロノームです
- bpmからmsに変換するには[expr 1000/($f1/60)]を利用します
- $f1は'All'キャンバスから送られた値を受け取る変数です
- [count]は0から任意の数値までカウントします
- 今回のシークエンサー(前のスライドを参照)では32個のボックス'p'があるため、0から31までカウントしています



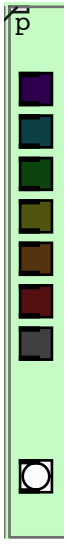


[select]について

```
select 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30 31
```

- **[select]** は入力端子から入力された値をselect内のすべての引数と比較し、対応する出力端子にbang[o]を出力します
- [select]の1番右の出力端子は、入力された値がどの引数にも合致しなかったとき、bang[o]を出力します

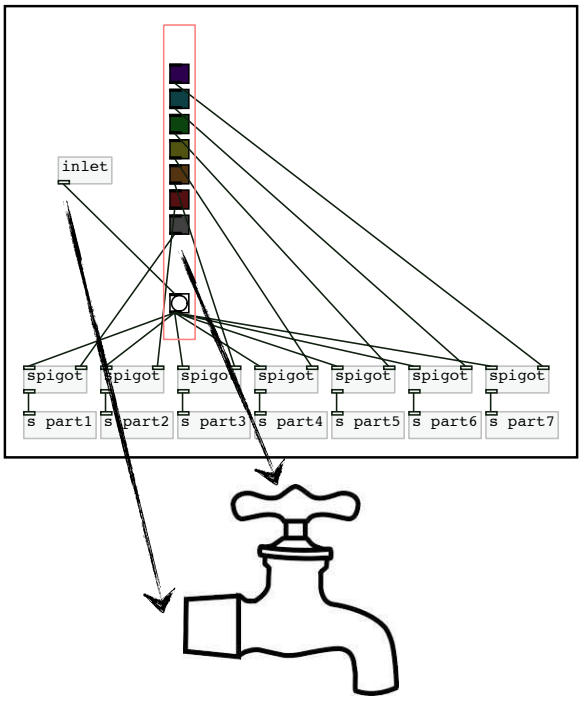
ビートと[p]について



- **[p]** は1ビート分の、各ビート上で音を再生するための抽象オブジェクトです
- それぞれのtoggleと楽器は一対一の関係です.
- ここではそれぞれ色付けしており、例えばキックドラムは灰色に、スネアは茶色になっています
- このbangは現在どのビートが再生されているか表示します

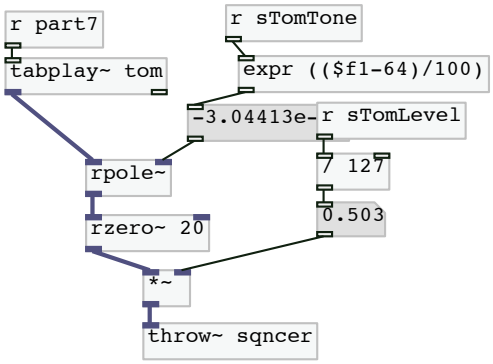
[spigot]について

- **[p]** の内部には[spigot]オブジェクトの集合体が隠れています
- **[spigot]** はメッセージを、右の入力端子の入力が0ではないときにだけ出力へ通します
- 右の入力端子の入力が0の時は、入力されたメッセージは遮断されます(入力を無視します)



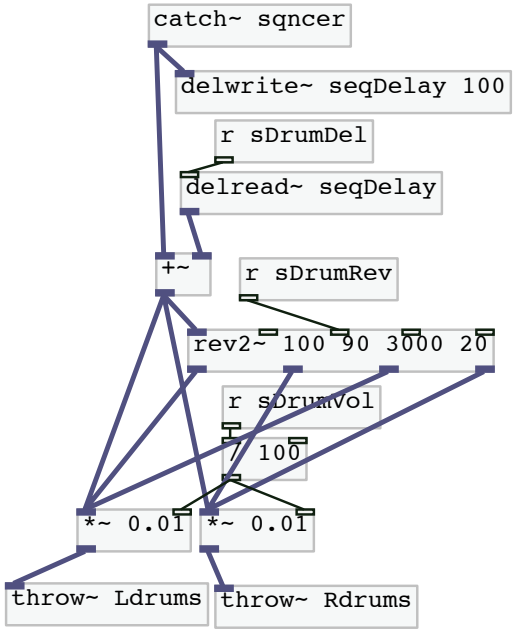
Playbackについて

- **[tabplay~]** によってそれぞれの楽器の配列が再生されます
- **[tabplay~]** はbangを受け取り、指定した配列の中に作成された音を出力します(ここでは'tom'がこれにあたります)
- 音の"tone"を変えることができます(基本的に1階のシェルビングフィルターです)
- 調整したレベルを**[throw~ sqncer]** へ送る前に、それぞれの楽器のレベルを調整します



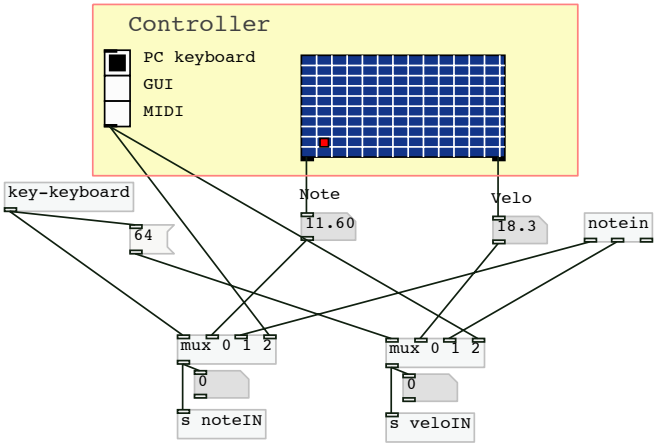
Playbackについて

- すべてのそれぞれの楽器の音は**[catch~ sqncer]** の中に追加されていきます
- **[delwrite~]** と **[delread~]** は音に遅延効果をかけます(ここでは遅延時間を100msに設定しています)
- **[rev2~]** は簡単な反響効果のオブジェクトです. ここでは反響効果の”ライブ感”をコントロールしています
- 最終的には[throw~] オブジェクトで異なるモジュールへ送られます



コントローラーについて

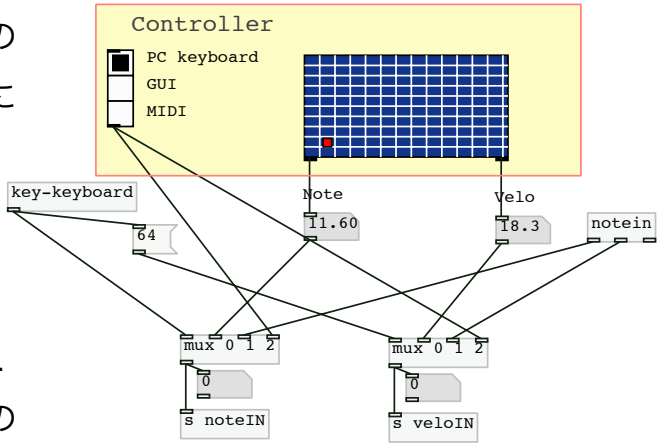
- コントローラーにはMIDIキーボードや、MIDIギター、マウス、PCのキーボードなどがあり、シンセサイザーのコントロラーとして使われることがあります
- 今回は、MIDIコントローラーとマウス、PCのキーボードでシンセサイザーをコントロールする方法を学びます



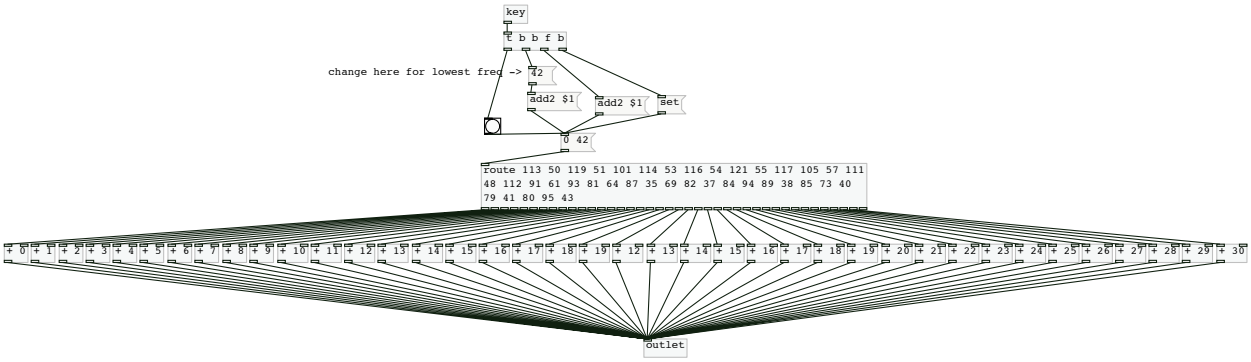
コントローラーについて

- **[hradio]** はクリックされたセルの番号を出力します. セルの数値は、0を起点に左から右に向かって大きくなっていきます

- **[mux]** は選択された入力端子を多重化してから出力します. 多重化する入力端子は1番右の入力によって選択します



キーボードコントローラー (PCの場合)

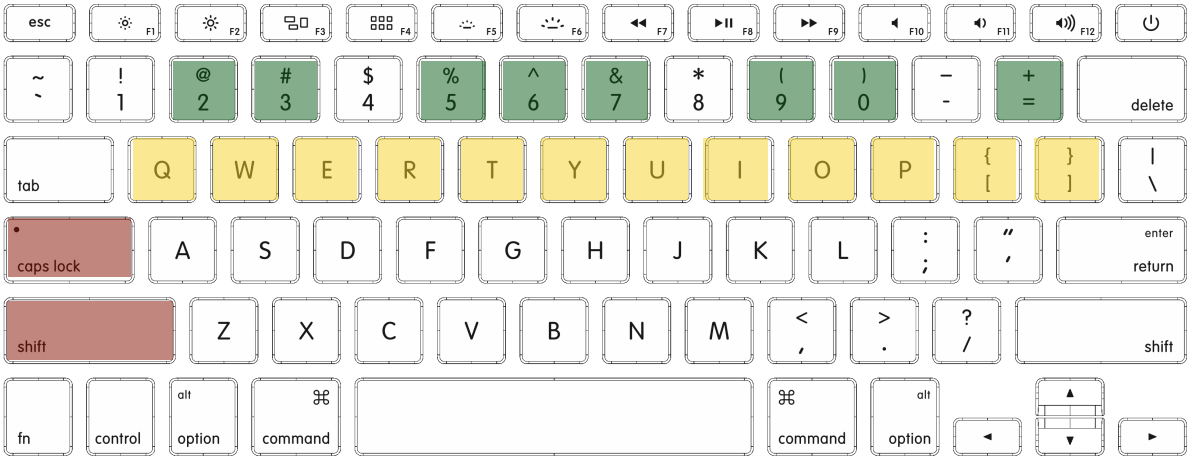


- キーボードのQから鍵盤のように弾くことができます(PCのキーボードをピアノに見立てて扱うことができます)

- ShiftキーまたはCAPSロックキーは音を1オクターブ上げるときに利用します

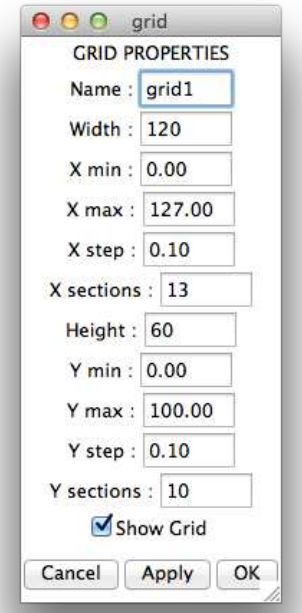
- 最も低い周波数(音)は、パッチ内部を編集することで変更することができます

キーボードコントローラーの有効範囲



GUIコントローラーについて

- GUIのコントローラーは**[grid]** で作成することができます
- **[grid]** はマウスの位置を追跡するためのXY領域を作成します
- ここでは、左の出力端子から0から127までのMIDI番号と、入力の速さ(例: 鍵盤を押すスピード)を出力します(ここでは0から100の範囲で出力します)
- 詳細設定は前回と同様の方法で行えます



MIDIコントローラー

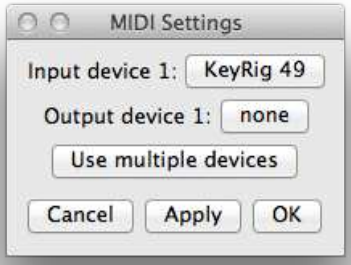
- コントローラーはUSBを使ってコンピュータと接続することができます
- コントローラーとコンピュータはMIDIコマンドを送受信しています
- MIDIはMusical Interface Digital Interfaceの略称です

Mac OSとMIDI

- 製造メーカーが提供するドライバーが必要な時があります. 利用する前にインストールしてください.
- 問題が発生した場合、Utilities > Audio MIDI Setup.app で解決できることがあります
- ショートカットキー⌘+2を利用することでMIDI Studioウィンドウを表示したり、隠すことができます

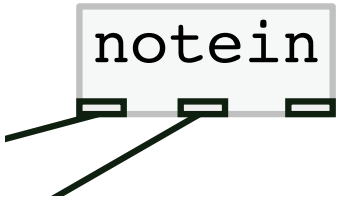
PdとMIDI

- MIDIデバイスの選択をするときはMediaメニューからMIDI Settingを選択してください
- 今回使用するM-Audio KeyRig 49は出力のみ対応しています(Pdでは入力デバイスとして設定します)
- デバイスの選択が終わったら“Apply”をクリックし“OK”でウィンドウを閉じます



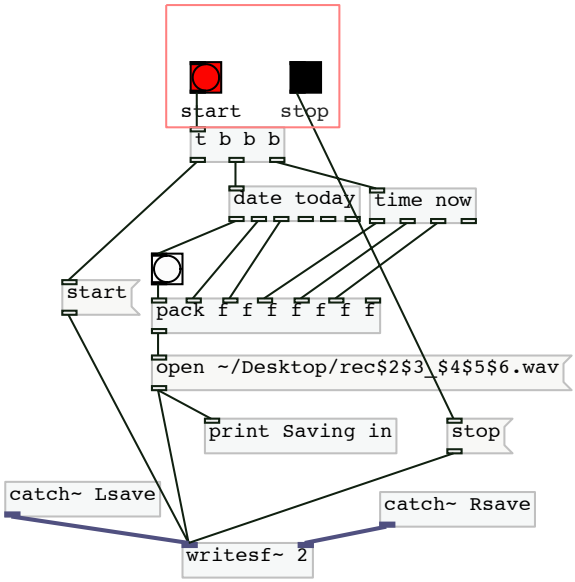
PdでMIDIを扱うときは

- [notein] はMIDIコントローラーで演奏されている音(出力している番号)を読み込み、出力します
  - 音の番号(0–127)
  - キーを押すときの速度の速さ (0–127)
  - チャンネル数 (1–16)
- 引数がないときはすべてのMIDIチャンネルから数値を読み取ります(オムニモード)



ファイルに音声を保存するとき

- **[writesf~ n]** はn個の音声ストリームをディスクに書き込みます
- **[pen X(**で場所を指定することで、どこに音声ファイルを保存するか設定できます
- **[start(** を入力することで録音を開始します
- **[stop(** で録音を停止することができます
- **[stop(** と、次の**[open X(** まで少し待つ必要があります



結論

- ここで紹介したことは、コンピュータを利用した音楽制作の一部にすぎません
- もっと詳しく知りたい方は、お気軽に以下のメールアドレスまでどうぞ!
  - julian@u-aizu.ac.jp

ありがとう！