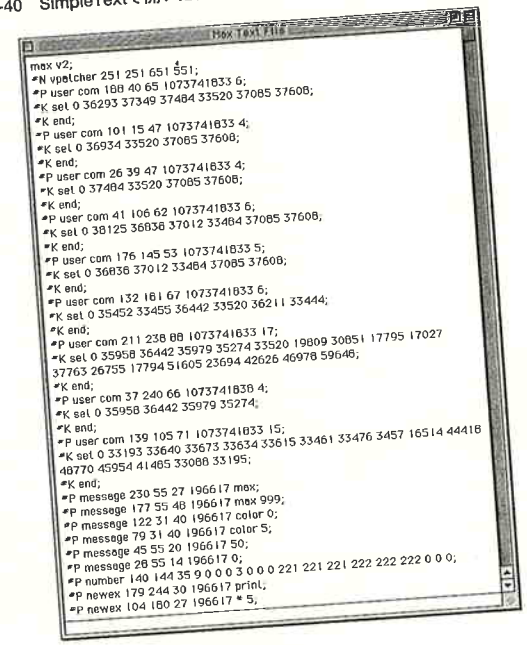
Max binary fileはMaxの独自フォーマットによるバイナリー形式で、Max text fileは一般的なテキスト形式となっている。Max binary fileはファイル容量が少なくて済むため、通常はMax binary fileとしてファイルを保存すればよいだろう。どちらの形式で保存しても、Maxで開けば同じパッチとして表示される。

テキスト形式で保存したファイルをSimpleTextなどのテキスト・エディターで開くと、記号や数字が並んでいることが分かる。また、MaxのFileメニューからOpen As Text File...を選べば、ファイル・フォーマットに関わらず、パッチをテキストとして開くことができる。このテキストはオブジェクトやパッチ・コードの接続などを表している。

現時点で、他のアプリケーションでMax text fileとして保存したファイルを利用する機会は少ないかもしれない。しかし、Maxが複数のプラットフォームで動作するようになれば、テキスト形式でパッチ・ファイルの交換が行われるだろう。また、電子メールの本文にパッチをテキスト形式で入れることも行われている。

■3-2-40 SimpleTextで開いたテキスト形式のパッチ・ファイル



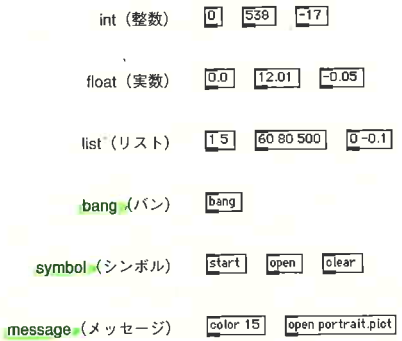
04/11/23

全部で何種類が
それらはどんな
メッセージか？
05/1/20

3-3 メッセージの種類

Maxではオブジェクトにメッセージを送ることで処理が行われる。したがって、どのような種類のメッセージがあり、それらをどのように使うかが重要になる。Maxで使用する基本的なメッセージのタイプは5種類だが、これらを組み合わせたメッセージも利用できる。ここでは、メッセージの種類と使用法について説明する。

■3-3-1 メッセージのタイプと例



bangメッセージ

bangメッセージは、オブジェクトに何らかの動作のきっかけを与える特別なメッセージだ。オブジェクト・パレットの4番目にあるbuttonオブジェクトは、それをクリックするとbangメッセージを出力する。また、メッセージ・ボックスにbangと入力してクリックしても、bangメッセージが出力される。

bangメッセージを受け取ったオブジェクトは、オブジェクトによって異なるが、何らかの動作を行うことになる。例えば、randomオブジェクトはbangメッセージを受け取ると、0からアーギュメントで指定した数値より1小さい数値の範囲で、ランダムに選ばれた整数を出力する。

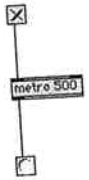
3-3-2のパッチでは、buttonオブジェクトかbangメッセージ・ボックスをクリックする度に、0から99までの範囲の数値がランダムにナンバー・ボックスに表示される。

■3-3-2 bangメッセージの利用

また、何らかの処理の結果としてbangメッセージを出力するオブジェクトもある。例えば、metroオブジェクトは、アーギュメントで指定した数値をmsec単位の間隔として、繰り返してbangメッセージを出力する。

3-3-3のパッチでは、toggle《Toggle》オブジェクトをクリックしてチェックすれば、metroオブジェクトが動作を開始する。metroオブジェクトは500msecごとにbangメッセージを出力し、bangメッセージを受け取ったbuttonオブジェクトが一瞬点灯する。したがって、このパッチは0.5秒ごとにチカチカとbuttonオブジェクトが点滅することになる。なお、toggleオブジェクトは、オブジェクト・パレットの5番目にある。

■3-3-3 bangメッセージを出力するmetroオブジェクト



ちなみに、bangは拳銃で弾丸を発射した音や、ドアや机を叩く音の擬音語だ。もちろん"バン"と読むのであって、"バング"ではない。

04/12/3
05/1/8 青

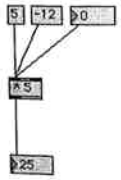
intメッセージ

intメッセージは、1つの整数をその内容とするメッセージだ。整数は、例えば、100、44100、0、−221といった具合に小数点以下の小数部を持たない整数部だけの数値である。内部的にはintメッセージは32ビットの符号付整数として表されるため、intメッセージの値としては、−2,147,483,648から2,147,483,647までの範囲となる。

intメッセージは、メッセージ・ボックスでは小数点を持たない整数として作成する。そのメッセージ・ボックスをクリックすれば、intメッセージが出力される。また、オブジェクト・パレットの6番目にあるナンバー・ボックスを用いてもよい。ナンバー・ボックスをクリックして上下にドラッグすれば、ナンバー・ボックスに表示される整数が変化し、intメッセージとして出力される。また、ナンバー・ボックスをクリックした上で、任意の整数をタイプしてenterキーを押してもよい。これでタイプした整数が表示されるとともに、アウトレットから出力される。

次の3-3-4の例では、メッセージ・ボックスをクリックするか、ナンバー・ボックスを操作すれば、その整数がintメッセージとして出力される。intメッセージを受け取った＊5オブジェクトは、その数値を5倍した値をintメッセージとして出力する。そして最後に、一番下のナンバー・ボックスがintメッセージを受け取り、乗算の結果である整数を表示することになる。

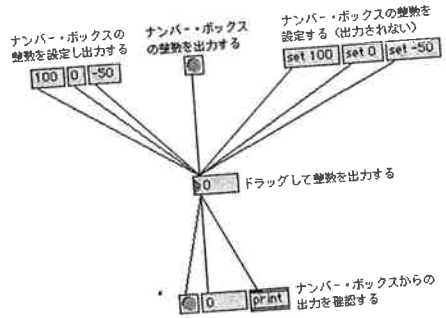
■3-3-4 intメッセージの利用



ナンバー・ボックスは、表示している整数を内部的に記憶している。そのため、bangメッセージをナンバー・ボックスに送ると、その整数が出力される。また、intメッセージをナンバー・ボックスに送ると、表示が変わるとともにその整数を出力する。

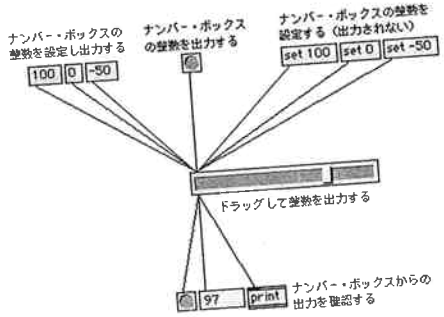
これに対して、setに整数を続けたメッセージを送ると、表示は変化するが、出力は行わない。3-3-5のパッチで、ナンバー・ボックスをドラッグして整数を変化させたあとに、buttonオブジェクトをクリックしてナンバー・ボックスから同じ整数が出力されることを確認してほしい。また、メッセージ・ボックスをクリックして、ナンバー・ボックスの表示や出力を比べて、intメッセージとsetメッセージの違いを理解しておこう。

■3-3-5 ナンバー・ボックスの動作



ナンバー・ボックスと同じことが、hsliderやdialオブジェクトについても言える。次のパッチは3-3-5の例とほとんど同じだが、hsliderの動作を確かめることができる。

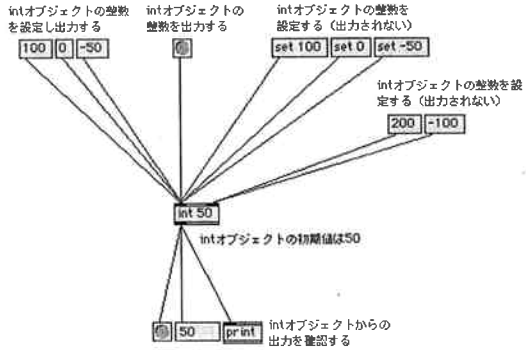
■3-3-6 hsliderオブジェクトの動作



ただし、オブジェクトの種類や設定によって、動作が異なることがある。例えば、デフォルト状態のhsliderは、0から127までの範囲の整数に限定される。したがって、0より小さい整数は0と見なし、127より大きい整数は127として扱われる。これらは、オブジェクトのインスペクターを使って設定することができる。

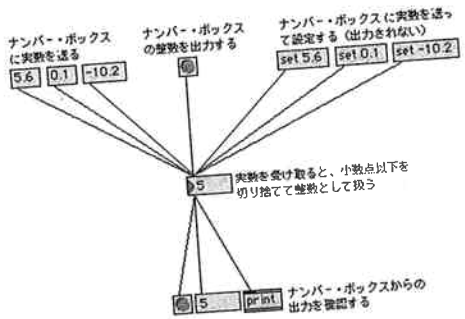
ユーザー・インターフェース・オブジェクトを必要としない場合は、intオブジェクトを整数の記憶や呼び出しのために用いる。intオブジェクトが、intメッセージを受け取れば、内部の値を変えるとともに出力も行う。setに整数を続けたメッセージは内部の値を変えるだけで、出力は行わない。そして、bangメッセージを受け取れば内部の整数を出力する。これらの動作はナンバー・ボックスと同じだと考えてよい。一方、intオブジェクトはアーギュメントによって初期値を決めることができる。また、第2インレットにintメッセージを受け取れば、出力は行わずに内部の値を変えるようになっている。

■3-3-7 intオブジェクトの動作



なお、ナンバー・ボックスやintオブジェクトがfloat（実数）メッセージを受け取った場合は、小数点以下を切り捨てて整数として扱うようになっている。これは、他のintメッセージを受け取るオブジェクトでも同じように処理される。

■3-3-8 実数を受け取ったナンバー・ボックスの処理



floatメッセージ

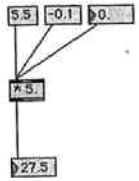
floatメッセージは、実数のメッセージで、実数は小数点以下の小数部を持つ数値だ。floatメッセージの例としては、3.14、5.、−0.006などを挙げることができる。floatメッセージは、内部処理としては32ビットの浮動小数点実数として扱われる。したがって、桁数が多い実数などでは誤差が生じるので注意が必要だ。また、floatメッセージの表記では3.25E+10のような指数表現は使用できない。

メッセージ・ボックスやオブジェクト・ボックスでの表現では、小数点を表すピリオドを付けるか否かで、intメッセージであるか、floatメッセージであるかが判断される。例えば、5はintだが、5. はfloatとなる。5. は5.0と表記する方が分かりやすい。ただし、多くのオブジェクトでは、5.0と表記しても、保存したパッチを開くと強制的に5. という表記になってしまう。

また、floatメッセージは、オブジェクト・パレットの7番目にあるフロート・ナンバー・ボックスを用いても作成できる。フロート・ナンバー・ボックスをクリックして上下にドラッグすれば、表示される実数が変化し、floatメッセージとして出力される。ここで、実数の整数部、すなわち小数点の左側をドラッグすれば整数部が変化し、小数点の右側である小数部をドラッグすれば小数部が変化する。つまり、最初にクリックする場所によって動作が異なる。フロート・ナンバー・ボックスをクリックした上で、任意の実数をタイプしてenterキーを押してもよい。これでタイプした実数が表示されるとともに、アウトレットから出力される。

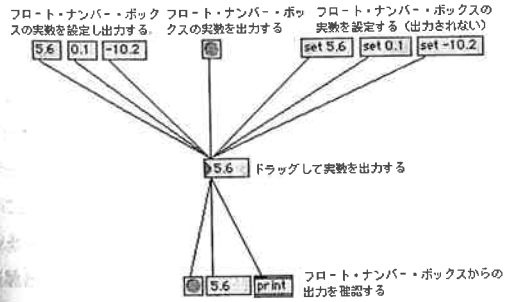
3-3-9の例では、メッセージ・ボックスをクリックするか、フロート・ナンバー・ボックスを操作すれば、その実数がfloatメッセージとして出力される。floatメッセージを受け取った * 5. オブジェクトは、その数値を5倍した値をfloatメッセージとして出力し、最後に一番下のフロート・ナンバー・ボックスが、乗算の結果を表示することになる。

■3-3-9 floatメッセージの利用



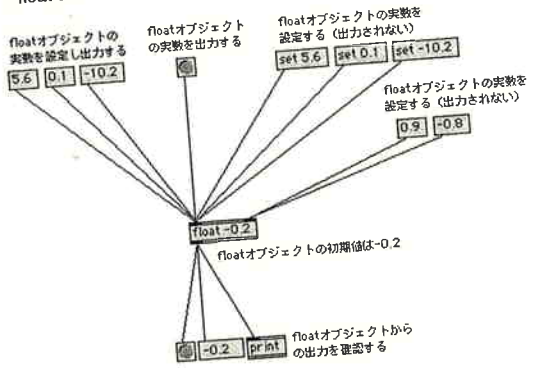
フロート・ナンバー・ボックスは、ナンバー・ボックスの実数版だと考えればよいだろう。フロート・ナンバー・ボックスは、表示している実数を内部的に記憶しており、bangメッセージを受け取ると、その実数を出力する。また、floatメッセージを受け取ると、内部状態と表示が変わるとともに、その実数を出力する。これに対して、setに実数を続けたメッセージを送ると、内部状態と表示は変化するが、出力は行わない。

■3-3-10 フロート・ナンバー・ボックスの動作



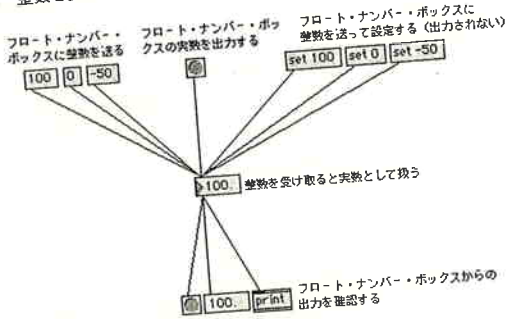
intオブジェクトの実数版に当たるのがfloatオブジェクトだ。扱うのが実数であることを除けば、使い方はintオブジェクトを同じと考えてよい。floatオブジェクトは内部に実数を記憶し、その値を変更し、出力することができる。

■3-3-11 floatオブジェクトの動作



フロート・ナンバー・ボックスやfloatオブジェクトが、intメッセージを受け取ると、その整数を実数と見なして処理を行う。つまり、10は10.0として扱う。このことは、他の実数を扱うオブジェクトでも同様だ。

■3-3-12 整数を受け取ったフロート・ナンバー・ボックスの処理



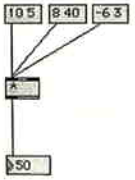
listメッセージ

listメッセージは、2つ以上の数値を半角スペースで区切って並べたメッセージだ。listメッセージの例としては、64 80 1000や0 −0.1、0.1 0.2 0.3などがある。ちなみに、数値以外のタイプの要素が含まれているときは、それはlistメッセージとは呼ばず、単にメッセージと呼ぶことになる。

特定のオブジェクトはlistメッセージの入力を必要とするが、そうでないようなオブジェクトでもlistメッセージが利用できる場合がある。例えば * オブジェクトは2つのインレットを持ち、それぞれのインレットに掛けられる数値と掛ける数値のメッセージを受け取る。ここで、2つの数値から成るリストを第1インレットに送れば、その2つの数値を掛け合わせた結果が出力される。これは、リストの最初の数値が第1インレットに、次の数値が第2インレットに送られたと考えればよい。

このように、いくつかの数値を複数のインレットに受け取るオブジェクトは、listメッセージによって1度に複数の数値を送れるようになっていることが多い。listメッセージをうまく使えば、メッセージ処理を簡素化できるだろう。

■3-3-13 listメッセージの利用

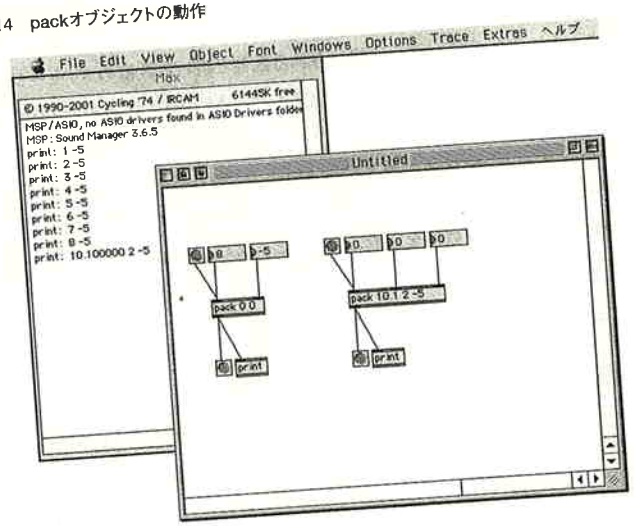


数値からlistメッセージを作るには、packオブジェクトを用いる。packオブジェクトはアーギュメントとしていくつかの数値を並べてリストの初期値を指定する。したがって、アーギュメントによってリストを構成する要素のタイプや個数が決まり、インレットの数も決まることになる。例えば、pack 0 0とすれば、初期値は0 0であり、整数2つから成るlistメッセージを出力する。pack 10.1 2 −5なら、初期値は10.1 2 −5であり、実数、整数、整数という並びのlistメッセージを出力することになる。

packオブジェクトのインレットには、後述するright-to-left orderのルールにより、右から順に数値を送り、第1インレットに数値やリストを受け取ったときにlistメッセージを出力する。第1インレットにbangメッセージを送ったときは、packオブジェクトが内部に記憶しているリストが出力される。

listメッセージを出力せずに、リストの最初の要素を設定したい場合は、setに数値を続けたメッセージをpackオブジェクトの第1インレットに送ればよい。

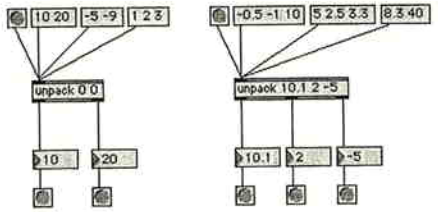
■3-3-14 packオブジェクトの動作



packオブジェクトとは逆に、listメッセージを受け取って、リストを構成する数値ごとに出力するためにunpackオブジェクトが用意されている。unpackオブジェクトもアーギュメントとしていくつかの数値やシンボルを並べてリストの初期値を指定する。このアーギュメントによってリストを構成する要素のタイプや個数が決まり、アウトレットの数も決まる。例えば、unpack 0 0とすれば、初期値は0 0であり、2つの整数から成るlistメッセージを受け取って、2つのアウトレットからリストの要素を出力する。

unpack 10.1 2 -5なら、初期値は10.1 2 -5であり、実数、整数、整数という並びの

■3-3-15 unpackオブジェクトの動作



listメッセージを受け取り、それぞれの要素を出力する。いずれもlistメッセージの最初の要素が第1アウトレットから、次の要素が第2アウトレットからという順に出力される。

アーギュメントの指定よりも少ない要素数のlistメッセージを受け取った場合は、リストの要素の数だけ出力される。要素数が多い場合、アウトレット数を超えた要素は無視され出力されない。bangメッセージを受け取ったときは、内部に記憶しているリストを各アウトレットから出力するが、unpackオブジェクトに対しては、setメッセージは利用できない。

なお、packやunpackオブジェクトは、アーギュメントにシンボルを指定すると、シンボルを要素とするメッセージを扱えるようになる。ただし、厳密には、listメッセージは数値だけで構成されるメッセージであり、シンボルが要素に含まれるメッセージはlistではない。それは単にメッセージと呼ぶ。

05/1/8

symbolメッセージ

symbolメッセージは、単純には文字列としてのメッセージだと考えればよい。helloやopen、startなどがsymbolメッセージの例として挙げられる。特定のsymbolメッセージは、オブジェクトへのコマンドとして機能し、オブジェクトに何らかの動作を指示する役目を持つ。どのようなsymbolメッセージを受け付けるかは、オブジェクトによって異なる。例えば、lcd《LCD》オブジェクトはビットマップ画像を描くオブジェクトだが、clearメッセージを受け取ると、表示している画像を消去する。lcdオブジェクト上でマウスを

ドラッグすると線画が描けるので、その後clearのメッセージ・ボックスをクリックすれば、画像が消去されることが分かるだろう。

同様にwritepictメッセージを送れば、ファイル保存ダイアログが開き、画像をファイルとして保存することができる。ちなみにlcdオブジェクトはオブジェクト・パレットの後半に、鉛筆のアイコンとして表されている。

例えばファイル名などのように、スペース(空白文字)を含む文字列を1つのシンボルとして扱いたい場合は、‘ と ’ で囲んで表記する。これらは英字フォントでの特殊文字だが、日本語キーボードでは ‘ はoptionキーを押しながら [キーを押し、’ はoptionキーとshiftキーを押しながら [キーを押せばよい。

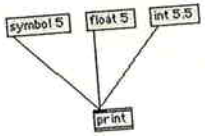
■3-3-16 symbolメッセージの利用

lcd オブジェクト

❷ メッセージのタイプ指定

メッセージ・ボックスは、通常はメッセージそのものを入力するだけでよい。ただし、メッセージの前にintやfloat、list、symbolを付ければ、そのメッセージのタイプを明示

■3-3-17 タイプを明示したメッセージ



的に指定することができる。このようなタイプの指定とメッセージとはスペースで区切って並べることになる。例えば、symbol 5というメッセージは、整数ではなくシンボルとしての5が出力されることになる。また、float 5というメッセージなら、実数の5.0が出力される。int 5.5なら、小数部が切り捨てられて、整数として5が出力される。3-3-17の例で、メッセージ・ボックスをクリックして、どのようなメッセージがMaxウィンドウに表示されるか、確認しておこう。

ほとんどの場合は、メッセージのタイプを明示しなくてもよい。しかし、特定のメッセージや複雑なメッセージ処理では、メッセージのタイプを明示的に指定する必要があるので、覚えておくとよいだろう。」

05/2/21