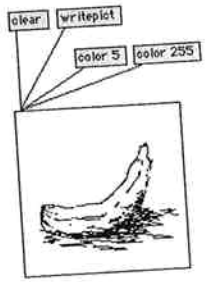
3-4 メッセージの処理

簡単なメッセージならば理解しやすいが、複数の要素から成る複雑なメッセージを用いなければならない場合も頻繁にある。ここでは、メッセージを構成する要素や、複雑なメッセージをどのように扱うかを説明する。また、メッセージを記憶し、メッセージを送る便利な方法についても触れてみよう。

複数要素のメッセージ

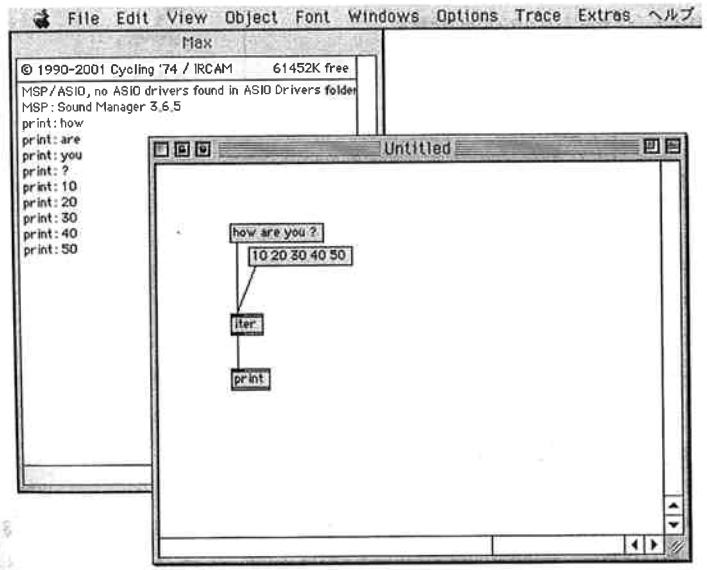
メッセージには、シンボルと数値から成るメッセージや、複数のシンボルから成るメッセージもある。このような複数の要素からなるメッセージは、それぞれのメッセージをスペースで区切ってメッセージ・ボックスに入力すればよい。このようなメッセージは、オブジェクトに対するアーギュメントを必要とするコマンドとして用いられることが多い。つまり、最初のシンボルがコマンドを表し、続く数値やシンボルがアーギュメントとして用いられる。例えば、lcdオブジェクトでは、描画する色をcolorメッセージで指定できるが、colorに続く0から255までの数値で色を指定することになる。3-4-1の例では、lcdオブジェクトにcolor 5メッセージを送れば、黄色で描かれるようになり、color 255メッセージなら黒色で描かれるようになる。この数値はMaxが持つカラー・パレットの色番号を意味している。

■3-4-1 複数の要素から成るメッセージ



listメッセージも含めて、複数の要素からなるメッセージはiterオブジェクトを使って、個々の要素ごとにメッセージとして出力することができる。この場合、先頭の要素（左側の要素）から順に出力される。

■3-4-2 iterオブジェクトによるメッセージの要素ごとの出力

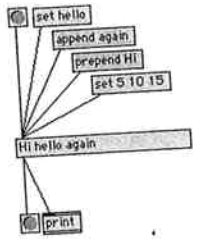


これまで見てきたように、メッセージ・ボックスには何らかのメッセージを設定して、メッセージ・ボックスをクリックすることで、そのメッセージを出力することができる。同じように、bangメッセージを受け取ったときも、メッセージ・ボックスの内容が出力される。

また、メッセージ・ボックスのメッセージを設定するには、setに続けてメッセージを送ればよい。例えば、set helloというメッセージをメッセージ・ボックスに送れば、メッセー

ジ・ボックスにはhelloと表示される。さらに、appendに続けてメッセージを送れば、メッセージ・ボックスの内容にメッセージを追加することができる。また、prependに続くメッセージを送れば、メッセージ・ボックスの内容の先頭にメッセージを追加することになる。いずれの場合も、メッセージ・ボックスの内容が変化するが、同時に出力されるわけではない。

■3-4-3　メッセージ・ボックスの出力と設定



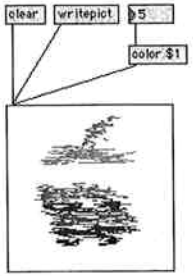
05/2/21

メッセージ中の$記号

メッセージ・ボックスに含めることができる特殊記号として$(ドル・マーク)がある。$は実際には数値を続けて$1のように用いる。$1はメッセージ・ボックスが受け取ったメッセージに置き換えられて出力される。例えば、color $1というメッセージ・ボックスに100というメッセージを送れば、$1が100に置き換えられるので、color 100というメッセージが出力されることになる。

そこで、3-4-4の例ではナンバー・ボックスをcolor $1のメッセージ・ボックスにつないでいる。そして、メッセージ・ボックスはlcdオブジェクトにつないでいるので、ナンバー・ボックスをドラッグすれば、描画色を指定することができる。このとき、colorメッセージのアーギュメントは0から255までが有効なので、ナンバー・ボックスのインスペクターで最小値を0に、最大値を255に設定しておくのがよいだろう。

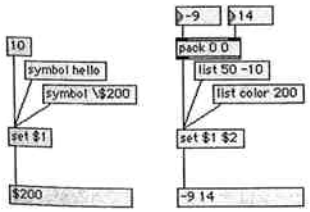
■3-4-4　メッセージ・ボックスにおける$記号の利用



なお、$1として置き換えるメッセージはシンボルであっても構わない。ただし、この場合のメッセージはsymbol fooのように、明示的にメッセージがシンボルであることを示さなければならない。また、set $1 $2のように複数の$記号を用いることもできる。ただし、メッセージ・ボックスは1つのインレットしか持たないので、$1と$2に当たるメッセージをリストとして送る必要がある。

また、$という文字そのものをメッセージとして扱いたい場合は、\$のようにバック・スラッシュ文字を$の前に付ければよい。バック・スラッシュ文字は、日本語フォントや日本語キーボードでは¥(円マーク)に相当する。

■3-4-5　より複雑な$記号の利用

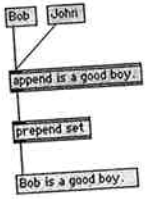


appendとprependオブジェクトによるメッセージ追加

複数の要素からなるメッセージを扱うときに、便利なのがappendオブジェクトとprependオブジェクトだ。appendオブジェクトは、受け取ったメッセージの後ろにアーギュメントを追加して出力する。これに対してprependオブジェクトは、受け取ったメッセージの前にアーギュメントを追加して出力する。アーギュメントはどのようなタイプでもよく、複数のアーギュメントを指定しても構わない。

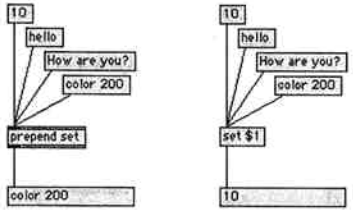
3-4-6の例では、Bobというメッセージを受け取ったappendオブジェクトは、アーギュメントのis a good boy.を後ろに付けてBob is a good boy.というメッセージを出力する。このメッセージを受け取ったprependオブジェクトは、アーギュメントのsetを前に付けて出力する。したがって、set Bob is a good boy.がメッセージ・ボックスに送られるので、メッセージ・ボックスにはBob is a good boy.と表示される。

■3-4-6　appendオブジェクトとprependオブジェクトの動作



appendオブジェクトやprependオブジェクトが便利なのは、受け取るメッセージの内容を気にしなくてもよい点だ。3-4-7の例の左側のパッチでは、どのようなメッセージを送っても、prependオブジェクトがsetを前に追加して出力するので、メッセージ・ボックスに表示することができる。これに対して、右側のパッチでは、set $1というメッセージ・ボックスを使っているので、数値メッセージはそのまま処理できるが、それ以外はエラーになってしまう。

■3-4-7　prependオブジェクトとsetメッセージの違い



sprintfオブジェクトによるメッセージ作成

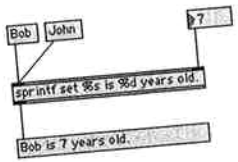
より複雑なメッセージを作成するためには、sprintfオブジェクトが用意されている。sprintfは、アーギュメントに%で始まる特別な記号を持ち、受け取ったメッセージを%記号の部分に置き換えて出力するオブジェクトだ。%記号はsやdといったアルファベットを用いて、受け取ったメッセージをどのようなタイプとして扱うかを指定する。これには、次のような種類がある。

%dまたは%ld	メッセージを整数として扱う
%f	メッセージを実数として扱う
%s	メッセージをシンボルとして扱う
%c	メッセージをASCIIコード番号として扱う

アーギュメントには複数の%記号を含めることが可能で、sprintfは%記号の数に応じたインレットを持つことになる。インレットの順序はアーギュメントの%記号の並びに対応する。したがって、第1インレットで受け取ったメッセージは、アーギュメントの最初の%記号に置き換えられ、第2インレットへのメッセージは、2番目の%記号に置き換えられる。

次の3-4-8の例では、sprintfオブジェクトはset %s is %d years old.というアーギュメントを持っているので、インレットは2つであり、第1インレットにシンボルを、第2インレットに整数を受け取ることになる。そこで、第2インレットに7を送り、第1インレットにBobを送ると、sprintfオブジェクトはset Bob is 7 years old.というメッセージを出力する。したがって、メッセージ・ボックスにはBob is 7 years old.と表示されるだろう。

■3-4-8 sprintfオブジェクトの動作

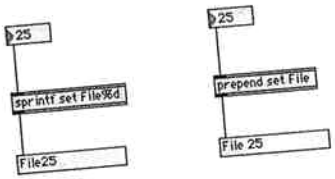


sprintfオブジェクトは複雑な構成のメッセージを作成するために便利だが、複数の要素を1つの要素としてまとめることができる点も重要だ。

3-4-9の例では、sprintf set File%dに25を送るとset File25が出力される。これに対して、prepend set Fileに25を送ればset File 25となってしまう。つまり、sprintfではFileと25は完全に連結され、File25という1つの要素になる。

一方、prependオブジェクトを用いるとFileと25はFile 25という2つの要素からなるリストになる。これはpackオブジェクトやsetメッセージも同様なので、2つの要素を1つの要素としてまとめるにはsprintfオブジェクトを使わなければならない。ちなみに、sprintfオブジェクトでもアーギュメントをFile %dのように指定しておけば、2つの要素のまま取り扱うことができる。

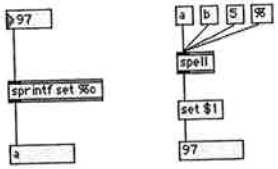
■3-4-9 sprintfオブジェクトとprependオブジェクトの違い



また、sprintfオブジェクトで%c記号を用いれば、受け取った整数をASCIIコード番号と見なして、それに対応する文字に置き換えて出力することができる。

なお、sprintfオブジェクトでは文字をASCIIコード暗号に変換することはできないが、spellオブジェクトを使えば実現できる。

■3-4-10 ASCIIコード暗号と文字との変換



複数のメッセージ

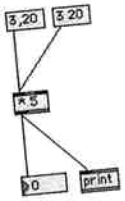
1つのメッセージ・ボックスに複数のメッセージを納めることもできる。このためには、個々のメッセージを , (カンマ)で区切る。それぞれのメッセージは左から順に連続的に出力される。例えば、3,20というメッセージ・ボックスをクリックすれば、まず3が出力され、次いで20が出力される。

3-4-11の例で、3,20と3 20の2つのメッセージ・ボックスの違いを確認しよう。3,20のメッセージ・ボックスをクリックすれば、まず3が出力され、* 5オブジェクトによって5倍されて15が出力される。次いで20が出力され、5倍されて100が出力される。これらは瞬時に行われるため、ナンバー・ボックスでは判別できないが、Maxウィンドウを見れば、printオブジェクトによって15と100が出力されたのが分かるだろう。

これに対して3 20はlistメッセージであり、メッセージ・ボックスをクリックすれば、1つのメッセージとして出力される。3 20を受け取った * 5オブジェクトは乗数を変更して3×20として乗算を行い、60を出力する。ちなみに、3 20メッセージを送ったあとに3,20メッセージを送れば、60と400が出力される。

なお、カンマという文字そのものをメッセージとして扱いたい場合は \, のようにバック・スラッシュ文字を , の前に付ければよい。バック・スラッシュ文字は、日本語フォントや日本語キーボードでは¥(円マーク)に相当する。

■3-4-11 複数のメッセージから成るメッセージ・ボックス

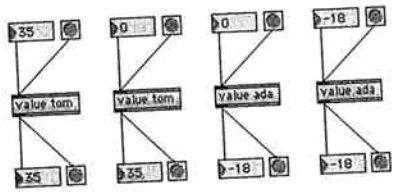


valueオブジェクトによるメッセージの記憶

すでに説明したように、ナンバー・ボックスやintオブジェクトを使って、メッセージを記憶することや取り出すことができる。ただし、オブジェクトによって扱えるメッセージの種類は限定されている。一方、valueオブジェクトを用いれば、いかなるメッセージでもオブジェクト内部に記憶し、それを呼び出すことができる。valueはvと省略してもよい。valueオブジェクトにメッセージを記憶させるには、メッセージをvalueオブジェクトに送るだけでよい。またvalueオブジェクトにbangメッセージを送れば、記憶しているメッセージが出力される。ただし、valueオブジェクトはアーギュメントとして何らかのシンボルを与えなければならない。そして、同じアーギュメントに持つvalueオブジェクトは、記憶するメッセージを共有する。

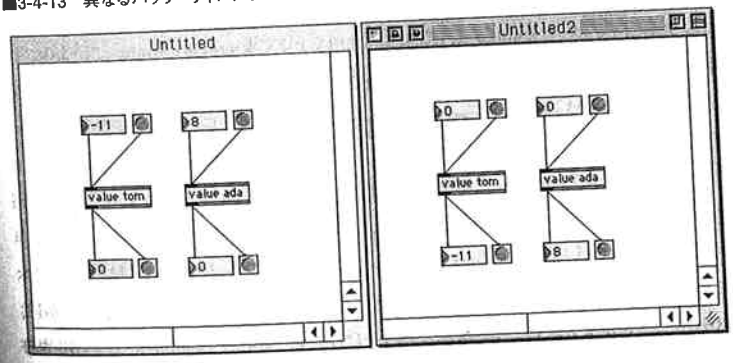
3-4-12のパッチで、どちらかのナンバー・ボックスをドラッグして、整数をvalueオブジェクトに送ってみよう。そして、buttonオブジェクトをクリックすると、valueオブジェクトからその整数が出力される。そして、同じアーギュメントのvalueオブジェクトにつながるbuttonオブジェクトをクリックすれば、同じ値が出力されることになる。さらに同じアーギュメントを持つvalueオブジェクトがあっても同じように動作し、異なるアーギュメントを持つvalueオブジェクトとは無関係に動作する。ここでは、intメッセージを扱っているが、他の種類のメッセージでも、valueオブジェクトは同じように動作する。また、途中でvalueオブジェクトへ送るメッセージの種類が変わっても構わない。

■3-4-12 valueオブジェクトの動作



ちなみに、一般的なプログラミング言語で言えば、valueオブジェクトは"型なしの変数"に相当する。valueオブジェクトのアーギュメントは変数名として考えればよい。つまり、valueオブジェクトはアーギュメントのシンボルで区別される箱を参照しており、その箱にメッセージを入れたり、取り出したりしているわけだ。また、異なるパッチ・ウィンドウにvalueオブジェクトがあっても、同じアーギュメントであれば同じ箱を共有することになる。つまり、変数のスコープ(有効範囲)は、Max全体となっている。そのため、不用意に同じアーギュメントのvalueを用いて混乱することがないように気を付けたい。

■3-4-13 異なるパッチ・ウィンドウでのvalueオブジェクトの動作



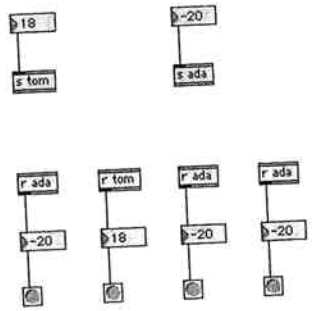
05/3/12

sendオブジェクトとreceiveオブジェクトによるメッセージ伝達

オブジェクトからオブジェクトへメッセージを送るためには、パッチ・コードでアウトレットとインレットを結ぶのが基本だが、sendオブジェクトとreceiveオブジェクトを用いれば、パッチ・コードなしにメッセージを送ることができる。例えて言うなら、sendオブジェクトはメッセージのワイアレス送信機で、receiveオブジェクトはワイアレス受信機になる。sendはs、receiveはrと省略することができる。

sendオブジェクトもreceiveオブジェクトも、アーギュメントに何らかのシンボルを指定する。そして、sendオブジェクトがメッセージを受け取ると、同じアーギュメントを持つreceiveオブジェクトへメッセージを送り、そのreceiveオブジェクトからメッセージが出力される。この際に同じアーギュメントのreceiveオブジェクトが複数あれば、それらはすべて同じメッセージを出力することになる。アーギュメントが異なるreceiveオブジェクトは何も出力しない。先の例で言えば、アーギュメントは送受信する周波数に相当し、周波数が一致するsendとreceiveオブジェクトとの間でメッセージが送られると考えればよいだろう。次のパッチでは、intメッセージで例示しているが、sendとreceiveオブジェクトはどのような種類のメッセージでも扱うことができる。

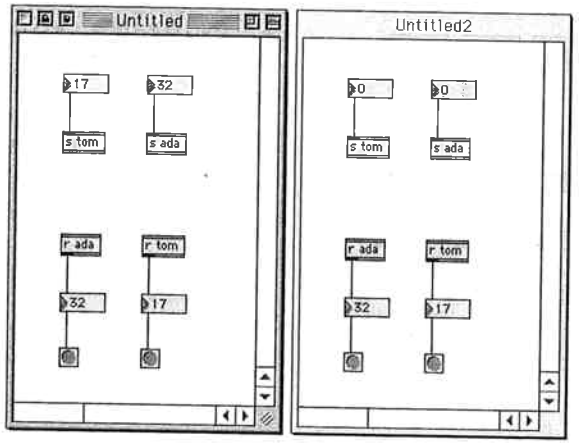
■3-4-14 sendオブジェクトとreceiveオブジェクトの動作



簡単なパッチでは、あえてsendとreceiveオブジェクトを用いる必要はない。しかし、パッチ・コードが入り組むような複雑なパッチになれば、sendとreceiveオブジェクトをうまく利用することで、すっきりとしたパッチを作ることができるだろう。

さらに、sendオブジェクトから送られたメッセージは、異なるパッチ・ウィンドウにあるreceiveオブジェクトでも受け取ることができる。これは、パッチ・ウィンドウ間でのメッセージのやり取りをするために利用できる。しかし、これは混乱の元になる場合もあるので、複数のパッチ・ウィンドウでsendやreceiveオブジェクトを用いるときには、十分に気を付けるべきだ。

■3-4-15 異なるパッチ・ウィンドウでのsendオブジェクトとreceiveオブジェクトの動作



このように、sendとreceiveオブジェクトは先に説明したvalueオブジェクトに似ているが、内部にメッセージを記憶するわけではない。むしろsendからreceiveオブジェクトへ目に見えないパッチ・コードがつながっていると考える方が妥当だろう。そのため、sendオブジェクトにbangメッセージを送った場合は、単にreceiveオブジェクトからbangメッセージが出力されることになる。また、sendオブジェクトがメッセージを受け取れば、ただちに対応するreceiveオブジェクトからメッセージが出力されるが、valueオブジェクトはbangメッセージを受け取ったときのみメッセージを出力する点も異なる。さらにvalueオブジェクトは内部にメッセージを記憶しているので、何度でもメッセージを取り出すことができる。

なお、メッセージ・ボックスを使って、receiveオブジェクトにメッセージを送ることがで