3-7 処理の実行順序

Maxは、オブジェクトとパッチ・コードから成るパッチとして、図形的なプログラミングを行う。このような方法は、分かりやすい反面、その動作原理を理解していなければ、混乱してしまうことになりかねない。そこで、ここではパッチがどのように実行されるか、その処理順序について説明する。これらを正確に理解することは、Maxでのプログラミングにおいて必須条件となる。

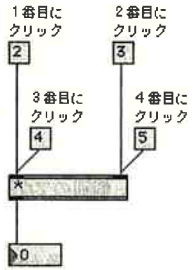
インレットの入力順序

すでに乗算のパッチで経験したように、複数のインレットを持つオブジェクトは、第2インレットにメッセージを受け取っても動作せず、第1インレットにメッセージを受け取ったときに動作を行う。これはright-to-left order(右から左への順序)と呼ばれ、パッチが動作する基本ルールになっている。

オブジェクトは、第2インレット以降にメッセージを受け取った場合は、内部的な状態を変えるだけで、表面的には何も変化が起こっていないように見える。つまり、第2インレット以降にメッセージを送っただけでは、パッチの処理は進行しない。これに対して、第1インレットにメッセージを受け取ったときに、オブジェクトは表示を変え、メッセージを出力するといった実際の動作を行う。例えば、* オブジェクトなら、第2インレットに数値を受け取ると、その数値を内部の乗数に記憶する。そして、第1インレットに数値を受け取ると、その数値と内部に記憶している乗数を掛けて、その結果をアウトレットから出力することになる。

ここで重要なのは、インレットにメッセージを送る順番だ。right-to-left orderに反して、左から右へとメッセージを送ってみよう。例えば、2×3という計算をする際に、まず、2のメッセージ・ボックスをクリックして * オブジェクトの第1インレットに2を送る。そして、そのあとで第2インレットに3を送ってみよう。ナンバー・ボックスに表示される計算結果は0のはずだ。次に、4×5という計算のために、第1インレットに4を送り、次いで第2インレットに5を送る。するとナンバー・ボックスの表示は12となり、期待していた結果とは異なってしまうことが分かるだろう。

■3-7-1 インレットへの入力順序のテスト用パッチ



なぜ、このような結果になるかは、先ほど説明したオブジェクトの動作を考えれば理解できる。

最初に、第1インレットに2を送ると、* オブジェクトはアーギュメントが省略されているので、乗数の初期値である0を掛けて、計算結果の0を出力する。次に、第2インレットに3を受け取ると、* オブジェクトは内部で記憶する乗数を3に設定するが、計算も出力も行わない。次いで、第1インレットに4が送られると、* オブジェクトは乗数の3を掛けて12という計算結果を出力する。これがナンバー・ボックスに表示される。そして、第2インレットに5を送ると、* オブジェクトの乗数が5になるが、計算や出力は行われない。このようにright-to-left orderに反した順序でメッセージを送ると、意図とは違う処理結果となってしまう。正しくは、2×3という計算なら、* オブジェクトの第2インレットに3を送り、その後に第1インレットに2を送る。

つまり、オブジェクトにメッセージを送る場合は、右側のインレットから左側のインレットに向かうようにメッセージを送らなければならない。これがright-to-left orderだ。もっとも、第2インレット以降にメッセージを送る必要がない場合は、第1インレットにメッセージを送るだけでよい。これでオブジェクトは何らかの動作を行う。

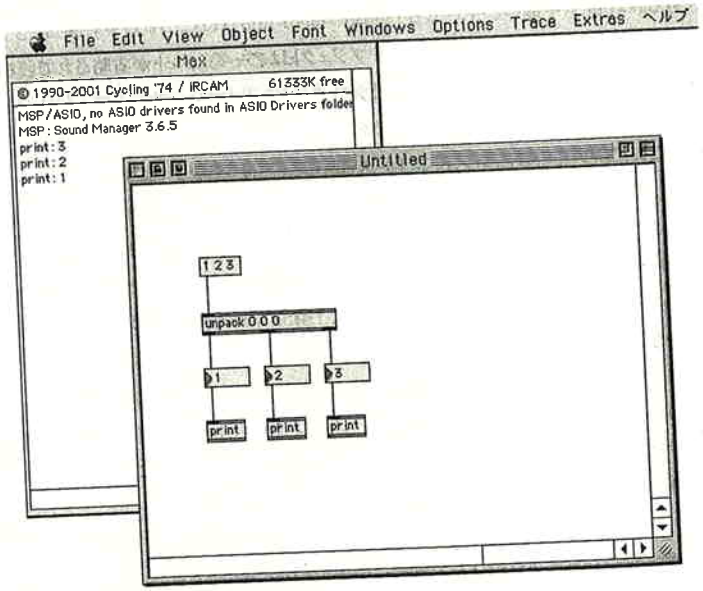
なお、この例のようにユーザーがオブジェクトを操作してメッセージを送る場合だけでなく、オブジェクトとオブジェクトをパッチ・コードでつなぐ場合も、right-to-left orderに従ってメッセージが流れるようにする必要がある。

アウトレットの出力順序

複数のアウトレットを持つオブジェクトがメッセージを出力する場合も、right-to-left orderに従った順序でメッセージが出力される。すなわち、最も右側のアウトレットから最初にメッセージが出力され、次いで左側に向かって順にアウトレットからメッセージが出力される。そして、最後に一番左のアウトレットからメッセージが出力される。

3-7-2の例はunpackオブジェクトを使って、メッセージの出力順序を調べている。unpackオブジェクトは、入力されるリストの要素のそれぞれを、個別のアウトレットから出力する。unpackオブジェクトのアーギュメントの数で出力する要素の数が決まり、その数だけのアウトレットを持つ。unpack 0 0 0とした場合は、3つの整数から成るリストを受け取り、それぞれの整数を順に3つのアウトレットから出力することになる。

■3-7-2 アウトレットからの出力順序のテスト用パッチ



メッセージ・ボックスの1 2 3をクリックすれば、そのリストがunpackオブジェクトに送られ、unpackオブジェクトはリストを要素に分解して、第1アウトレットから1を、第2アウトレットから2を、そして第3アウトレットから3を出力する。これはナンバー・ボックスで確認できるが、瞬間的に出力されるので順序までは判別できない。

そこで、printオブジェクトによってMaxウィンドウに表示された内容を見ると、3、2、1の順序で出力されたことが分かる。つまり、最初に第3アウトレットから出力され、次いで第2アウトレット、最後に第1アウトレットから数値が出力されたわけだ。これはright-to-left orderに一致している。ちなみに、ナンバー・ボックスは受け取った数値を表示すると同時に、その数値をアウトレットから出力するようになっている。

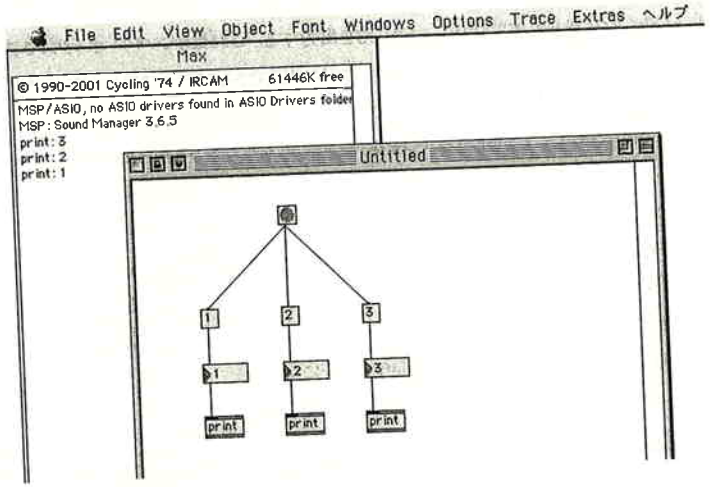
このように、オブジェクトが複数のアウトレットからメッセージを出力する際は、右側のアウトレットから左側のアウトレットに向かう順序でメッセージを出力する。これがアウトレットからの出力順序におけるright-to-left orderになる。

位置による実行順序

1つのアウトレットから、複数のインレットにパッチ・コードがつながっている場合は、どのような順序でメッセージが送られるだろうか？ 3-7-3の例では、buttonオブジェクトをクリックすると、パッチ・コードにbangメッセージが送られる。bangメッセージを受け取ったメッセージ・ボックスは、その内容である数値を出力する。数値を受け取ったナンバー・ボックスは、その数値を表示するとともに、同じ数値を出力する。そして数値を受け取ったprintオブジェクトが、その数値をMaxウィンドウに表示する。このパッチでは、このような一連の処理が3カ所で行われている。

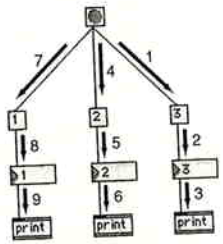
このように、1つのアウトレットから複数のパッチ・コードによって、それぞれ別のオブジェクトへメッセージが送られる場合も、right-to-left orderが適用される。つまり、最も右側にあるオブジェクトへ最初にメッセージが送られ、次いで左側にあるオブジェクトにメッセージが送られ、最後に最も左側のオブジェクトにメッセージが送られることになる。言い換えれば、同時に実行されようとする複数のオブジェクトがある場合、右側から先に実行される。これがオブジェクトの実行順序におけるright-to-left orderのルールとなる。

■3-7-3 位置による実行順序のテスト用パッチ



これまでの例でも同じだが、あるオブジェクトがメッセージを受け取ると、それ以降パッチ・コードでつながっている一連のオブジェクトを次々とメッセージが流れていく。そして、メッセージを出力しないオブジェクトに達するまで連続して処理が行われる。したがって、buttonオブジェクトは、まず3のメッセージ・ボックスにbangメッセージを送ると、終端のprintオブジェクトまで連続してメッセージが流れていく。その次にbuttonオブジェクトは2のメッセージ・ボックスにbangメッセージを送る、といった順序で処理が行われる。このパッチでのメッセージが流れる順序を図式すると、次のようになる。

■3-7-4 メッセージが流れる順序

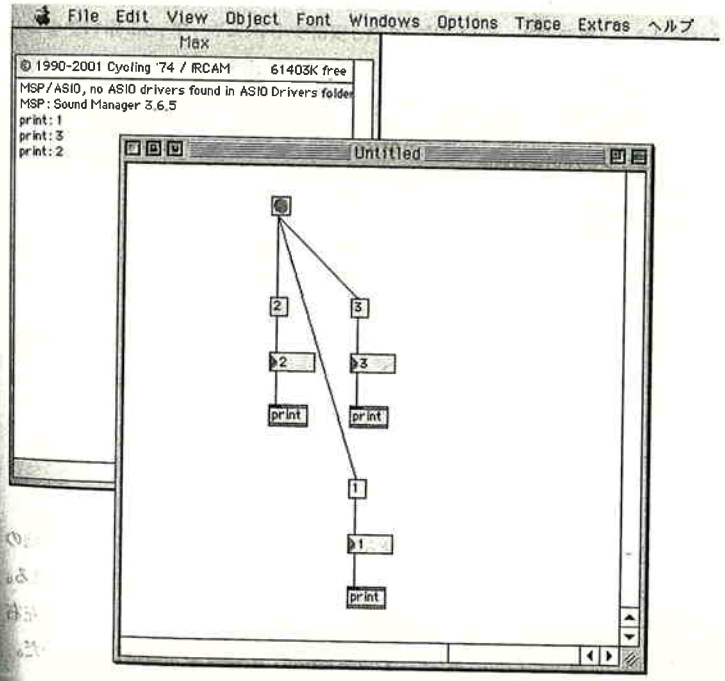


では、オブジェクトの左右の位置がまったく一緒の場合は、どうなるだろうか？ 次の例では、メッセージ・ボックスの3と1とは左右の位置が同じになっている。この場合は、right-to-left orderのルールだけでは実行順序を確定できない。

そこで、第2のルールとしてbottom-to-top orderが適用される。すなわち、同時に実行されようとする複数のオブジェクトの左右位置が同じ場合、より下側のオブジェクトから順に実行される、というルールだ。

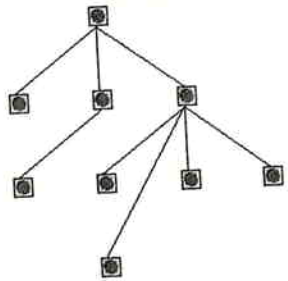
したがって、この例では、最初に下側の1にbangメッセージが送られ、次に3にbangメッセージが送られる。そして最後に左側にある2にbangメッセージが送られることになる。Maxウィンドウの表示を見れば、1、3、2という順序で表示されており、bottom-to-top orderのルールに従っていることが分かる。

■3-7-5 左右位置が同じ場合の実行順序のテスト用パッチ



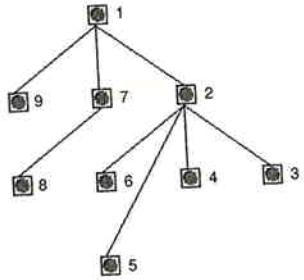
実用性はないが、3-7-6の例でオブジェクトが実行される順序を考えてみよう。一番上のbuttonオブジェクトをクリックすれば、すべてのbuttonオブジェクトが点灯するが、これはどのような順序で実行されているのだろうか？

■3-7-6　オブジェクトの実行順序のテスト用パッチ



この例で、オブジェクトが実行される順に番号を付ければ、以下のようになる。

■3-7-7　オブジェクトの実行順序



もう1度まとめれば、同時に実行されようとするオブジェクトが複数ある場合、右側のオブジェクトから順に実行される。左右位置が同じであれば、下側から実行される。これがオブジェクトの実行順序のルールとなる。ちなみに、複数のオブジェクトが左右も上下も同じ位置で、ぴたりと重なっている場合には、明確な優先順位はないようだ。

triggerオブジェクトによる実行順序の指定

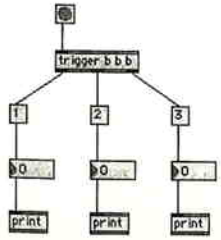
先に説明したように、Maxではオブジェクトの実行順序が、オブジェクトの位置関係で決まる。これはright-to-left orderとbottom-to-top orderというルールに従って決定されるので、オブジェクトの配置を調べれば、どのように実行されるかが分かる。

しかし、複雑で込み入ったパッチになれば、オブジェクトの実行順序を直観的に判断できないであろう。また、正しく動作していたパッチであっても、不用意にオブジェクトの位置関係を変えてしまい、その結果正しく動作しなくなるかもしれない。

そこで、同時に実行されようとする複数のオブジェクトの実行順序を明示的に指定する方法がある。それはtriggerオブジェクトを用いて行う。triggerオブジェクトは、アーギュメントにbやiといった記号を指定し、それらの記号の数だけのアウトレットが作成される。bはbangであり、iはintを表している。そして、triggerオブジェクトがメッセージを受け取ると、アーギュメントで指定したタイプにメッセージを変換し、それぞれのアウトレットからメッセージを出力する。このとき、アウトレットの出力順序はright-to-left orderに従う。triggerはtと省略することもできる。

3-7-3の例を、triggerオブジェクトを使って修正すれば、次のようになる。このパッチでは、buttonオブジェクトをクリックすると、bangメッセージがtriggerオブジェクトに送られる。triggerオブジェクトはアーギュメントに従って、3つのアウトレットからそれぞれbangメッセージを出力する。このとき、right-to-left orderによって右から左へと順に各アウトレットからbangメッセージが出力される。その結果、3、2、1という順にオブジェクトが実行されることになる。

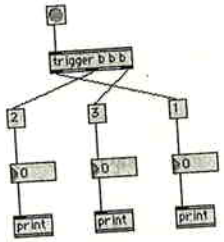
■3-7-8　triggerオブジェクトによる実行順序の指定



これだけでは先の例と同じだが、triggerオブジェクトを用いれば、オブジェクトの位置関係を変えても実行順序は変わらない。次の例でも、同じように3、2、1という順にオブジェクトが実行される。

この場合、triggerオブジェクトがアウトレットから出力する順序だけでオブジェクトの実行順序が決まるのであって、位置関係によって決める必要がないからだ。

■3-7-9 位置関係を変えても実行順序は変わらない



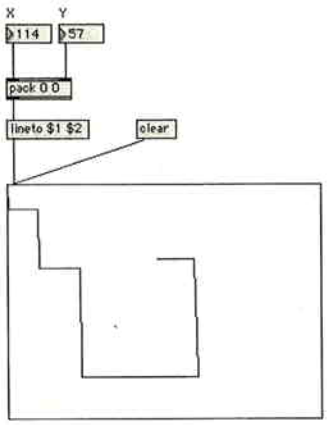
triggerオブジェクトのアーギュメントに指定できる記号には、bやi以外に、floatを表すfやlistを表すl、symbolを表すsを用いることができる。また、特定の数値やシンボルを定数として指定してもよい。triggerオブジェクトは、受け取ったメッセージを、アーギュメントで指定したタイプに変換して出力する。

例えば、ビットマップ画像を描くlcdオブジェクトは、linetoメッセージによって、現在位置から指定した座標位置までの線を描く。

次の3-7-10のようなパッチで、ナンバー・ボックスを用いて座標を指定すると、X座標を変えた場合は線が描かれるものの、Y座標を変えても線が描かれない。packオブジェクトは受け取った数値をリストにして出力するオブジェクトだが、第2インレットに数値を受け取っただけでは出力を行わない。そして、第1インレットに数値が入力されると、リストを出力する。

したがって、Y座標を変えても線は描かれない。ちなみに、これらの座標はlcdオブジェクトの左上隅を原点(0,0)とするピクセル単位で表されている。

■3-7-10 lcdオブジェクトを使った直線の描画パッチ



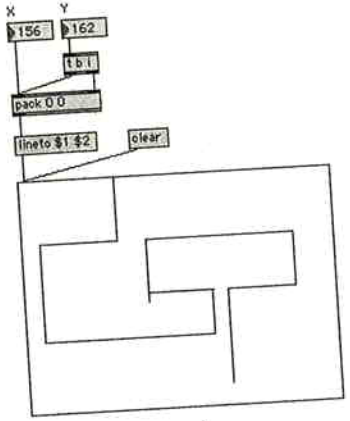
これを解決するには、packオブジェクトがbangメッセージを受け取ると、内部状態をリストとして出力する働きがあることを利用する。

trigger b iというオブジェクトを作成し、Y座標のナンバー・ボックスをtriggerオブジェクトにつなぐ。そしてtriggerオブジェクトのアウトレットとpackオブジェクトのインレットが対応するようにパッチ・コードをつなぐ。これで、Y座標のナンバー・ボックスを動かしたときも直線が描けるようになる。triggerオブジェクトが第2アウトレットから数値を出力し、次に第1アウトレットからbangメッセージを出力するからだ。ちなみに、triggerはtという省略形を用いている。同様の処理はbondoオブジェクトを用いても可能だ。

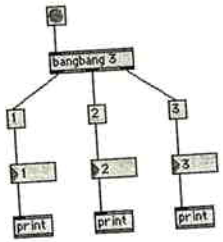
なお、実行順序を決めるために、出力するのがbangメッセージだけでよいならば、3-7-12のパッチのようにbangbangオブジェクトも利用できる。

bangbangオブジェクトは、アーギュメントでアウトレットの数を指定し、何らかのメッセージを受け取ると、右側のアウトレットから順にbangメッセージを出力する。bangbangはbと省略することもできる。

■3-7-11 改良した直線描画パッチ



■3-7-12 bangbangオブジェクトによる実行順序の指定



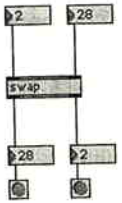
swapによるメッセージの入れ替え

swapオブジェクトまたはfswapオブジェクトを使って2つのメッセージを入れ替えて出力することができる。swapはintメッセージを扱い、fswapはfloatメッセージを扱う点が異なるが、いずれも処理は同じである。

これらのオブジェクトは2つのインレットと2つのアウトレットを持っており、まず、第2インレットに受け取った数値をオブジェクトの内部に記憶する。そして、第1インレットに数値を受け取ると、その数値を第2アウトレットから出力し、続いて内部に記憶している数値を第1アウトレットから出力する。つまり、見かけ上、2つの数値を入れ替えるような動作になる。

3-7-13のパッチでは、上部のナンバー・ボックスを操作すれば、メッセージが入れ替わって出力されることが分かる。

■3-7-13 swapオブジェクトの動作



■3-7-14 swapとパッチ・コードのクロスによる違い