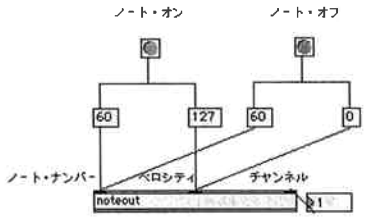
4-4 コンピューター・キーボードによるMIDI演奏

MIDIコントローラーやMIDI音源モジュールなどのMIDI機器は、コンピューターに接続するさまざまな入力装置、出力装置の1つとして考えることができる。そうであれば他の入力装置によってMIDI機器をコントロールすることもできるはずだ。ここでは、最もポピュラーな入力装置としてコンピューター・キーボードを取り上げ、これによりMIDI機器をコントロールし、演奏するパッチを作ってみよう。

ノート・メッセージを作る

ノート・メッセージは、ノート・オンとノート・オフからなり、それぞれ"音を鳴らせ!""音を止めろ!"という意味を持ったMIDIメッセージだった。Maxでは受信用オブジェクトとしてnoteinオブジェクト、送信用オブジェクトとしてnoteoutオブジェクトが用意されており、パッチ内でノート・メッセージはノート・ナンバー(音高、0〜127)、ベロシティ(強度、0〜127)、チャンネル(1〜16)という3つの要素として取り扱われることもすでに説明した。逆に言えば、ノート・メッセージを作り、noteoutオブジェクトから外部MIDI機器に送信するためには、この3つの要素をMaxメッセージとしてnoteoutオブジェクトに与えることになる。4-4-1のパッチ例でノート・オンとコメントを付けた左側のbuttonオブジェクトをクリックすると、外部MIDI音源が中央ドのキーで発音を始めるだろう。持続系の音色であれば音は鳴りっぱなしになるはずである。

■4-4-1 ノート・メッセージを作る



続いてノート・オフとコメントを付けた右側のボタンをクリックすると鳴っていた音が止まる。ここでベロシティが0に変わっただけで、あとの数値はすべて同じだということをもう1度確認しておいてほしい。

コンピューター・キーボードで演奏する

この方法を応用し、コンピューター・キーボード(以下、単にキーボードという)でMIDI音源を演奏するパッチを作ってみることにしよう。4-4-2のパッチではキーボードからの入力を取り込むオブジェクトとしてkeyおよびkeyupを使っている。

keyオブジェクトはキーを押した瞬間に、keyupオブジェクトは離した瞬間に、そのASCII値をパッチ内部に取り込むオブジェクトである。ASCII値とは文字1つ1つに割り当てられた世界標準のコード番号で、例えばAは65、Bは66と決められている。ここではこの数値をそのままノート・ナンバーとして利用する。またkeyオブジェクトはキーを押す、keyupオブジェクトはキーを離すという両方のイベントを区別して取り込むので、ちょうどMIDIコントローラーの鍵盤で演奏するようにノート・オン、ノート・オフのきっかけ(トリガー)として利用することができる。

gateはその名の通りゲートの開閉によりメッセージを通したり遮断したりするオブジェクトだ。左インレットに0を受け取るとゲートを閉じ、0以外の数値を受け取るとゲートを開いて、右インレットから入ってくるメッセージを通過させる。toggle《Toggle》オブジェクトはクリックを受けると1と0を交替で出力し、それに対応してボックス上に×印を表示するため、オン/オフ・スイッチとしてよく用いられる。ここではゲートの開閉をスイッチングしている。このようにgateオブジェクトを挟む理由は演奏以外のときにkeyやkeyupオブジェクトが出力するメッセージを遮断しておくためである。

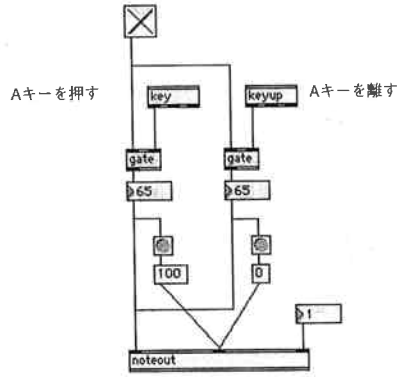
さて、実際にキーボードを使って演奏してみよう。まず、コンピューターの入力メニューが英数になっていることを確認する。次にキーボードのcapslockをonにする。こうしてキーボードのAキーを押してみる。capslockをonにしているので大文字のAをタイプしたことになる。AにはASCII値として65が割り当てられているので、keyオブジェクトは整数メッセージ65を出力する。toggleオブジェクトがonの場合、65はgateオブジェクトを通過し、ナンバー・ボックスに値を表示させながらbuttonオブジェクトに渡される。buttonオブジェクトは何らかのメッセージを受け取るとすべてそれをbangメッセージに

変換する。bangメッセージを受け取ったメッセージ・ボックスは、アーギュメントとして書き込まれた100をnoteoutオブジェクトの第2インレットに渡す。一方、メッセージ65はナンバー・ボックスから枝分かれし、直接noteoutオブジェクトの第1インレットにも渡される。これでnoteoutオブジェクトはベロシティ100、ノート・ナンバー65を受け取り、MIDI音源に対してチャンネル1でノート・オンを送信する。

次にキーボードのAキーから指を離すとkeyupオブジェクトが対応するASCII値65を出力し、以下同様な過程を経て、noteoutオブジェクトにベロシティ0、ノート・ナンバー97が渡されチャンネル1でノート・オフメッセージを送信する。

capslockをonにした理由は、大文字A～Zに割り当てられたASCII値が65～90、小文字a～zに割り当てられたASCII値が97～122のため、そのままノート・ナンバーに当てはめた場合に、大文字の方が音域として適当だからである。

■4-4-2　キーボードで演奏するパッチ(1)



音の鳴りっぱなし

このパッチでは滅茶苦茶にキーをタイプしていると音が鳴りっぱなしのまま止まらないトラブルが発生することがある。MaxでMIDIを扱う場合に1度は出会うトラブルがこ

の音の鳴りっぱなし現象だろう。これはノート・オンに対応するノート・オフがうまく伝えられなかったときに発生するトラブルだ。

パッチに戻ってどのような場合に音が鳴りっぱなしになるのかを検証してみよう。例えばcontrolキーを押しながらZキーを押したとする。この場合、keyオブジェクトは26を出力するのでノート・ナンバー26、ベロシティ100、チャンネル1のノート・オンが送信される。これでZキーから指を離せば音は止まるはずだ。

ところがcontrolキーを先に離してからZキーを離した場合、keyupオブジェクトが出力するのは26ではなく90になる。なぜならZキーを離したときすでにcontrolキーは解放されており、keyupは単に英文字ZのASCII値を出力するからだ。その結果、ノート・ナンバー90、ベロシティ0、チャンネル1のノート・オフが送信されてしまうのである。すなわち先に送信したノート・オンに対応すべきノート・オフが、ノート・ナンバーの異なる別物に変わってしまったわけだ。キーボードを滅茶苦茶タイプしたときに起こる音の鳴りっぱなしは、どうやらこうした原因によるものと推理できる。

強制的にノート・オフを送信する

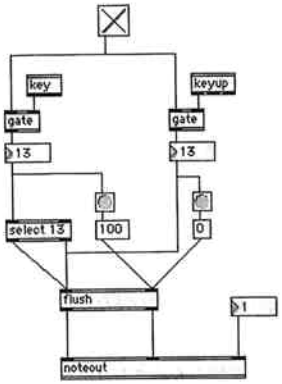
4-4-3のパッチは、このトラブルを解決するために、音の鳴りっぱなしが起こったときにreturnキーを押すことで強制的に音を切る仕組みを付け足した改良版である。flushというオブジェクトは左右のインレットに受け取る数値をそのまま通過させると同時に、特殊な形で内部に一時的に格納する。例えば左右のインレットに受け取る数値の組を仮にx、y(ただしx、yとも0ではない)とすると、次にx、0の組を受け取るまでx、yのペアはflushオブジェクト内部に一時格納される。この状態で左インレットにbangメッセージを受け取ると、flushオブジェクトは内部に格納してあったx、yをx、0に変えて出力する。これをnoteoutオブジェクトと組み合わせると、ホールド中のノートに対していつでも強制的にノート・オフを作り出すことが可能になる。

selectはアーギュメントとして書き込まれた数値と同じものを左インレットから受け取ると左アウトレットからbangメッセージを出力し、それ以外のものを受け取ると右アウトレットから通過させるオブジェクトだ。ここではアーギュメントとして13と書かれているのでASCII値が13の改行コード、つまりreturnキーを押すとselectはbangメッセージを出力し、それをflushオブジェクトに渡すことになる。returnキー以外のキーを押せばそ

のASCII値のまま通過させる。

先ほどの例をもう1度使うと、controlキーを押しながらZキーを押したときにflushは26、100をそれぞれ受け取り、そのまま通過させると同時に、ペアにして内部に一時格納する。音が鳴りっぱなしのトラブルが発生したのでユーザーがreturnキーを押すとselectオブジェクトはbangメッセージを出力し、それを受け取ったflushは右アウトレットから0、左アウトレットから26を出力してめでたく音を切ることができるというわけだ。

■4-4-3 キーボードで演奏するパッチ(2)



makenoteオブジェクトによるノート・オフの生成

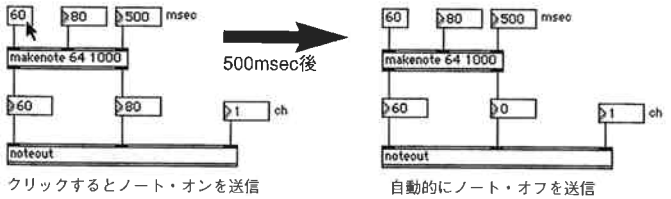
何らかの原因でノート・オフが失われてしまい音が鳴りっぱなしになるトラブルは、ノート・メッセージがノート・オンとノート・オフという2つのメッセージで構成され、しかもそれぞれが時間をおいて送信される以上、リアルタイムにMIDIを扱う場合に避けて通れない問題だと言える。しかし今、説明したflushオブジェクトによる解決は初めて触れる人にとっては難解に映るかもしれない。別の解決法としてMaxではmakenoteというオブジェクトが用意されている。これを利用すればノート・オフの問題を一切気にすることなくプログラミングを進めていくことができる。

もう1度ノート・オンとノート・オフの関係を考えてみよう。MIDIでは音の高さをノート・ナンバーで表し、音の強さをベロシティで表している。ところが音楽で言うところの音値つまり音の長さに当たるものはMIDIにはない。リアルタイムに機器をコントロールするMIDIでは、ノート・オンが送信されたあとに時間をずらせてノート・オフを送信することで、結果的に音の長さを表現しているのだ。これは特に演奏者の立場からすると自然なことである。現在鳴らしているこの音をいつ止めるのかは演奏者のその場の判断であり、音の長さをあらかじめ正確に決めることはできないからだ。しかし別の立場から見れば、音を高さ、強さ、長さの3要素で考える方が自然な場合も多いだろう。

makenoteは、ノート・ナンバーとベロシティ、そして音の長さであるデュレーションを与えることで適切なタイミングで自動的にノート・オフ用のメッセージを作ってくれるオブジェクトだ。デュレーションとは持続時間を指し、msec単位で音の長さを指定する(1msecは1,000分の1秒)。

次のパッチのようにmakenoteは3つのインレットを持ち、受け取った数値をノート・ナンバー、ベロシティ、デュレーションとして解釈する。アーギュメントとして2つの数値を書き込むことができるが、これはベロシティとデュレーションの初期値であり、これらの初期値はインレットから別の数値を受け取れば内部で置き換えられる。したがって、このパッチの状態ではベロシティが80、デュレーションが500msecとなっている。

■4-4-4 makenoteオブジェクトによるノート・オフの自動生成



クリックするとノート・オンを送信　　自動的にノート・オフを送信

さて、ここで60と書き込んだナンバー・ボックスをクリックすると、makenoteオブジェクトは右アウトレットから80、左アウトレットから60を出力し、それを受けてnoteoutオブジェクトはノート・ナンバー60、ベロシティ80、チャンネル1のノート・オンを送信する。