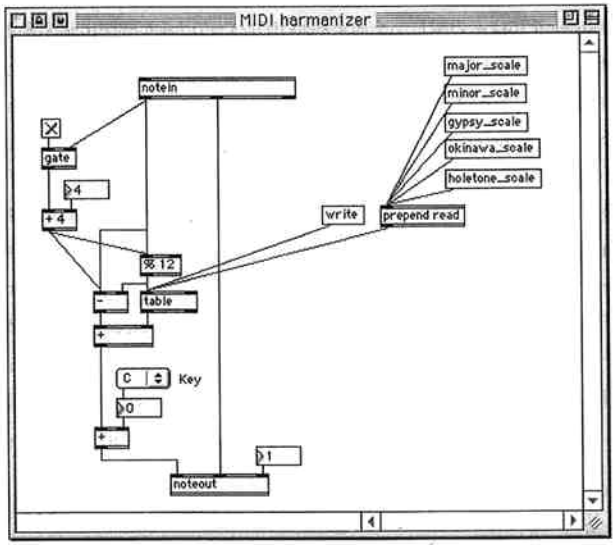
■4-5-11 スケール変換を行うMIDIハーモナイザー



4-6 MIDIディレイ

ある条件が"真"ならば処理Aを実行し、"偽"ならば処理Bを実行するという条件処理、あるいは、ある条件が"真"の間は処理Aを繰り返すというループ処理はあらゆる言語のプログラミングにおいて重要な制御構造となる。音楽をやるのになぜ条件処理やループ処理の話をしないといけないのか？と疑問に感じるかもしれないが、これは特別なことではない。例えばもしMIDIコントローラーの真ん中から左でサンプラーを鳴らし、右で別の音源を鳴らすキーボード・スプリットのパッチを作ったとしたら、それは条件処理を実行したことになるのだ。一方、ループ処理についてはMaxではまさにパッチ上でループ回路を描くことにより表現される。ここではサンプル・パッチとして簡単なMIDIディレイを作りながら条件処理やループ処理の表現をまとめて紹介しよう。

シンプルなMIDIディレイ

MIDIディレイとは、MIDIコントローラーからのノート・メッセージを受け取り、それを一定の時間間遅らせて送信するもので、4-6-1はそのパッチ例である。

noteinオブジェクトによって受信されたノート・メッセージは、そのままnoteoutオブジェクトから送信されると同時にpipeオブジェクトにも渡されている。pipeというオブジェクトは受け取ったメッセージを設定した時間（msec単位）遅らせて出力する。アーギュメントとして、受け取るメッセージ・タイプを表した数値を必要なインレットの個数だけ書き込み、ディレイ・タイムの初期値を続ける。ここでは0 0 250と書かれているが、最初の0 0は整数としてメッセージを受け取るインレットを2個作ることを意味する。もちろん整数であれば0 0でなく1 5であっても構わない。なぜなら一応初期値として1や5が設定されはするものの、実際にはstripnoteオブジェクトからのメッセージによってすぐに初期値は内部で書き換えられてしまうからだ。ここでは個数とタイプが大切なのであって数値の大きさは問題にはならない。

アーギュメント最後の250はディレイ・タイムの初期値で、第3インレットから新しいディレイ・タイムを受け取るまでは第1、第2インレットから受け取る数値を250msec遅らせて出力することを意味する。

4-7 乱数と確率による自動作曲プログラム

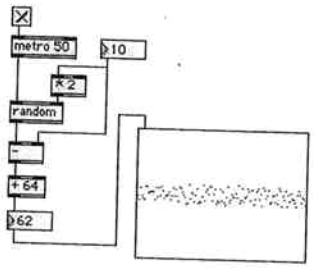
コンピューターが音楽に使われるようになり、アルゴリズミック・コンポジションという耳慣れない言葉が使われるようになった。これは、音選びの規則をプログラム化して行う作曲という意味になる。森を散策していて突然神の啓示のごとく頭の中に音楽が浮かぶ天才作曲家にしても、寝ることも食べることも忘れてひたすらピアノと五線譜に向かう努力派作曲家にしても、ある音の次にどんな音が来るべきかその都度選び取っていることに変わりはない。この音選びをコンピューターにやらせようというのである。この音選びの規則は、音高に次々と2を足していくといった単純なものでもいいし、数学や物理学の難解な理論を応用したものでも構わない。そしてこの1つとしてランダムに選ぶという方法論がある。例えば、サイコロを振って出た目で音を決めるというのも音選びの規則に違いない。ジョン・ケージ(John Cage)は易経に基づき偶然に音を選ぶことで作曲を行った。ヤニス・クセナキス(Iannis Xenakis)は、数学の確率論や分子運動のふるまいをモデルに偶然性を計算し音選びを行った。音楽の中で偶然性を取り入れる試みはこれまでたくさん実行されている。ここでは、人間は直接に音を選ぶのではなく、音選びの規則を考え、実際にはコンピューターがランダムに音を選ぶ簡単な自動作曲プログラムを作ってみることにしよう。

最も一般的なrandomオブジェクト

ランダムに音を選ぶ自動作曲プログラムに進む前に、Maxでランダムネスを扱ういくつかのオブジェクトや方法をまとめておこう。randomは、その名の通り乱数を生成する最も一般的なオブジェクトで、左アウトレットにbangメッセージを受け取ると、アーギュメントとして書き込んだ数値、あるいは右インレットに入力した数値の範囲内で整数の乱数を出力する。例えばアーギュメントとして128と書き込めば、0～127の範囲でランダムに整数を出力する。一口に乱数と言ってもさまざまな種類があるが、randomオブジェクトが出力する乱数は一様乱数と呼ばれ、すべての出力数値の発生確率が等しいものだ。例えばrandom 5は0～4の範囲で乱数を出力するが、何万回もbangを与え続け、0、1、2、3、4のそれぞれの出力回数を数えると、どれもほぼ等しくなる。

4-7-1のパッチはmetroオブジェクトが規則正しくbangメッセージを出力し、それによりrandomオブジェクトが乱数を出力するものだが、中央値64を中心にして乱数のバラツキの度合いを調整できるようにしてある。4-7-1のようにナンバー・ボックスを10にすると、randomオブジェクトは0～19までの乱数を出力するが、10を減算されるので－10～9の範囲となり、最終的に中央値64が加算され54～73の範囲の乱数になる。また出力される数値の動きを表示するために、ここではmultiSlider《MultiSlider》オブジェクトを使っている。

■4-7-1 randomオブジェクトの振る舞い



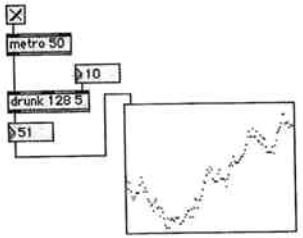
multiSliderオブジェクトは、整数だけでなく実数も取り扱うことができ、一般には複数のスライダーを持った入力インターフェースとして使われるが、Get Info...の設定を次のようにすると、受け取った数値の軌跡を順に表示させることができる。設定は、Slider Range の最小値を0、最大値を127とし、取り扱う数値のタイプをIntegerつまり整数とし、Slider StyleをPoint Scrollにしている。

■4-7-2 multiSliderオブジェクトのインスペクター

千鳥足のdrunkオブジェクト

drunkは少々風変わりな乱数生成オブジェクトだ。randomオブジェクトと異なり、このオブジェクトは自分が出力した直前の数値を記憶してそれとの関係で次の数値を作り出す。その結果、酔っぱらいが千鳥足で歩くようなフラフラした動きが生み出される。アーギュメントとして乱数の範囲、歩幅の大きさを初期値として指定する。第3インレットにナンバー・ボックスをパッチングしてあるのは、動作中に歩幅の大きさを変更できるようにするためだ。multiSliderオブジェクトの表示がrandomオブジェクトとは異なっているのが分かるだろう。

■4-7-3 drunkオブジェクトの振る舞い



確率テーブル

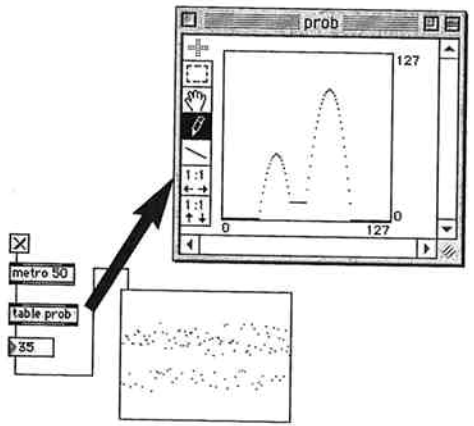
ランダムな出来事、予測不可能な出来事は、必ずしもrandomオブジェクトが出力する一様乱数のように均等な確率で発生するのではなく、確率に偏りがある場合の方が多いかもしれない。「4-5 スケール変換MIDIフィルター」(P371)の中で紹介したtableオブジェクトは、確率テーブルという特殊な用法により確率に偏りがある乱数発生オブジェクトとして使用することができる。特殊な用法といってもインレットに整数やリストを与える代わりにbangメッセージを与えてやりさえすればtableは乱数発生オブジェクトに変身してくれるのだ。ただアドレスとその値についての考え方がこれまで見てきた用法と異なる点で特殊なのである。

確率テーブルにおいてはtableウィンドウに描かれるグラフは確率分布曲線だと考えることができる。ここではアドレスはアウトレットから出力されるべき数値となり、アドレスの値はその確率の大きさになる。出力されるものがこれまでのtableオブジェクトの用法と逆転していることに注意してほしい。

アーギュメントとしてprobと名前を付けたtableオブジェクトにおいてtableウィンドウの内容は2つの山を持つ曲線になっている。もちろんこの曲線は鉛筆ツールでフリーハンドで描くこともできるし、何らかの数式パッチを作って描かせることもできる。この曲線はアドレスが40付近と80付近に確率の高まりがある確率分布曲線で、table probは、bangメッセージを受け取るとこの確率分布に従ってランダムにアドレスを出力する。

具体的に見てみよう。例えばこのテーブルでアドレス(x軸の値)が0から27まで、および98から127まではアドレスの値(y軸の値)が0になっている。このことは確率が0であることを示しており、table probに何万回bangメッセージを与えても0～27、98～127の数値は出力されないことになる。それ以外のアドレス28～97についてはtable probがbangメッセージを受け取る度にランダムに出力されるが、どのアドレスが選ばれやすいかという確率は均等ではなく、80付近の数値が最も頻繁に出力されやすく、次に40付近の数値がそれに続く。multiSliderオブジェクトの表示もそれを反映して2つの帯状の形になっている。

■4-7-4 確率テーブルとしてのtableオブジェクトの振る舞い

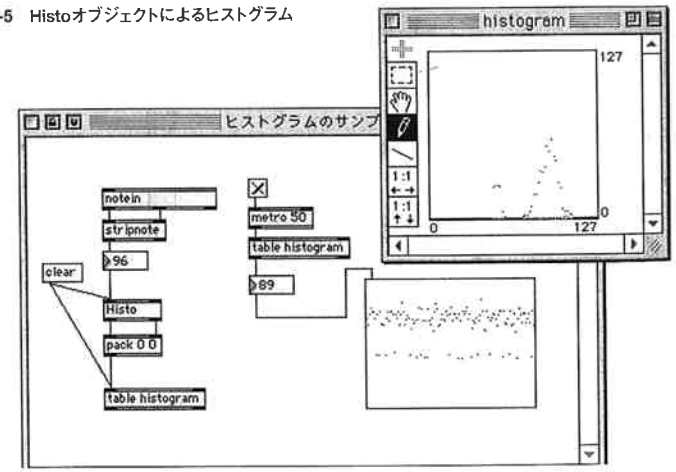


ヒストグラム

前述の確率テーブルでは、確率の分布をユーザー側が任意に描くものだった。しかし、現実の出来事をつかまえて、それらが何回発生したのかを数えることで、現実を反映した確率の分布をtableオブジェクトに与えることもできる。この出来事と出来事の発生回数をグラフ化したものをヒストグラムと呼び、MaxではHistoというオブジェクトがそのために用意されている。Histoオブジェクトは左インレットに整数を受け取ると、そのまま左アウトレットから通過させると同時に、右アウトレットから過去にその数値を受け取った回数を出力する。clearメッセージを受け取ると回数のカウントはリセットされ、また最初からカウントを始める。

4-7-5のパッチでは、noteinオブジェクトで受信されたノート・メッセージはstripnoteオブジェクトでノート・オフをカットされ、ノート・オンのノート・ナンバーだけがHistoオブジェクトに渡される。Histoオブジェクトは受け取ったノート・ナンバー(x)とその回数(y)を出力するので、それをpackオブジェクトによりリストにまとめ、histogramと名付けたtableオブジェクトに渡す。tableオブジェクトはx、yのリストを受け取ると、アドレスxに値yを格納する。こうしてノート・ナンバー60の鍵盤を何度も弾くと、tableオブジェクトのアドレス60の値がどんどん増加していくことになる。

■4-7-5 Histoオブジェクトによるヒストグラム



同じパッチ内にはもう1つのtable histogramがあり、先ほどの確率テーブルと同じパッチングが施されているが、アーギュメントが同一のtableオブジェクトは、パッチ上に何個置かれても、すべてその内容が共有される。つまりHistoオブジェクトによって左側のtable histogramの内容がいつ書き換えられても、その内容は右側のtable histogramに即座に反映するのだ。こうしてMIDIコントローラーで演奏を続けていると、tableオブジェクトは、実際に演奏されたノート・ナンバーを、その発生確率に従ってランダムに出力するようになり、コンピューターまかせではなく、その場のインタラクティブ性を加えた乱数を得ることができる。

ランダム作曲プログラム

さて、以上の内容をふまえて、いよいよ乱数と確率による自動作曲プログラムを作ってみよう。4-7-6のパッチはノート・ナンバー、ベロシティ、デュレーション、トリガーのタイミングすべてを、コンピューターがランダムに決定するものだ。

まずパッチの最上部は、確率テーブルを用いてトリガーのタイミングをコントロールしている部分だ。table timingの内容は、アドレス0、1、2、4、8、16のところにのみ値が書き込まれており、bangメッセージを受ける度にこのうちのいずれかを出力する。確率の大きさはパッチ例のようにアドレス0と2が大きくなっている。

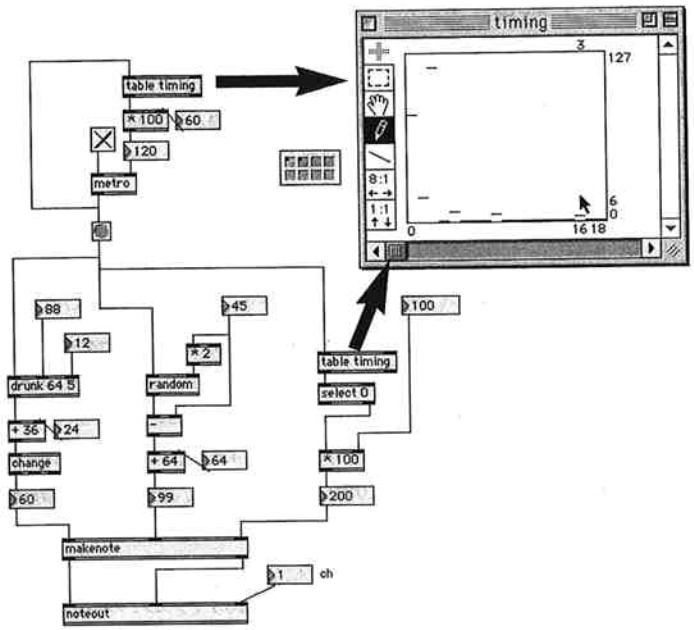
tableオブジェクトの出力は * で乗算され、それがmetroオブジェクトの時間間隔となっている。toggleオブジェクトをonにするとmetroオブジェクトは動作し始める。本来metroオブジェクトは設定した時間間隔で規則正しくbangメッセージを出力し続けるものだが、ここではbangメッセージを出力するごとに新たな時間間隔の値を受け取るので、不規則な時間間隔でbangメッセージを出力することになる。

このbangメッセージ1つが、1個のノート・メッセージをランダムに生成する。つまり最初にパッチ右側でデュレーションの値、次に中央部でベロシティの値、最後に左側でノート・ナンバーの値がそれぞれランダムに決定され、makenoteオブジェクトに渡される。

デュレーションはtable timingの確率テーブルを利用して決定される。table timingは0を出力する可能性があり、これは音の長さとしてのデュレーションには不適当である。selectはアーギュメントとして書き込んだ数値と同じものを受け取れば左アウトレットからbangメッセージを、それ以外の数値を受け取れば右アウトレットからそのまま出

tableオブジェクト

■4-7-6 ランダム作曲プログラム



力してくれるので、それを利用して0をカットしている。

ベロシティは、先ほど紹介したrandomオブジェクトの使用法により、中央値とそこからのバラツキという2つのパラメーターで決定される。

ノート・ナンバーについては、先ほど紹介したdrunkオブジェクトの使用法にオフセット値(最低値)を加算して決定される。changeオブジェクトは、新しく受け取った数値と直前に受け取った数値を比べ、異なっていれば出力し、同じであれば何も出力しない。drunkオブジェクトは連続して同じ数値を出力する可能性があり、その結果、同じノート・ナンバーのノート・メッセージが極めて短い時間間隔で送信される可能性が出てくる。そうなるとMIDI機器が不自然な発音を引き起こす場合があるので、それをchangeオブジェクトで防止している。

このパッチには多数のナンバー・ボックスがあり、さまざまなパラメーター設定を変更することで、音楽の表情を大きく変えれるようになっている。このように多数のパラメーター設定のセットを一括してパッチとともに保存し、瞬時に呼び出すためにはpreset《Preset》というオブジェクトが便利だ。パッチ・ウィンドウがロック状態で、shiftキーを押しながらpresetオブジェクト上の小さなボックスのいずれかをクリックすると、ナンバー・ボックスをはじめとするインターフェース系のオブジェクトの現在の状態を一括して記憶させることができる。記憶したボックスには左上に黒い点が打たれる。パラメーター設定のセットを呼び出すには、この黒い点が打たれたボックスをクリックすればよい。このようにして異なったパラメーター設定のセットを次々と別のボックスに記憶させ、いつでも呼び出すことができ、これらはパッチとともに保存される。

このランダム自動作曲パッチにより生み出される音楽は、人間が規則を与え、コンピューターがランダムに音選びをした結果だ。パラメーターの設定によってはいかにもたどたどしく、およそ音楽的には聞こえないかもしれない。また別のパラメーター設定では生々しく音楽的に躍動したものに聞こえるかもしれない。音楽的とはどういうことか?を考えるきっかけを与えてくれそうなパッチである。