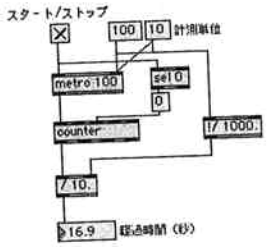
ところで、metroとcounterオブジェクトを使ってもストップ・ウォッチが実現できると考えてもおかしくはない。ただし、同じ処理をmetroとcounterオブジェクトを用いて実現すれば3-12-21のようになり、やや入り組んだパッチとなる。clockerオブジェクト使用版の方がスマートな処理となっていることが分かる。

■3-12-21　metroとcounterによるストップ・ウォッチ



もちろん、これはclockerとmetroのオブジェクトとしての優劣を表しているわけではない。ストップ・ウォッチを作る場合にはclockerオブジェクトの方が適しているというだけだ。他の用途ではmetroオブジェクトの方が便利なこともあるだろう。このようにMaxには同じような動作を行うオブジェクトがいくつか用意されているが、作成するパッチの目的や用途に応じて、適切なオブジェクトを選ぶことが重要になる。

ちなみに、周期的な動作を行うオブジェクトには、これら以外にtempoオブジェクトがある。tempoオブジェクトは動作間隔をテンポ(BPM)で指定し、1小節に対する分割数を指定するようになっている。音楽的なパッチを作成する場合にはtempoオブジェクトを用いるのがよい場合もあるだろう。

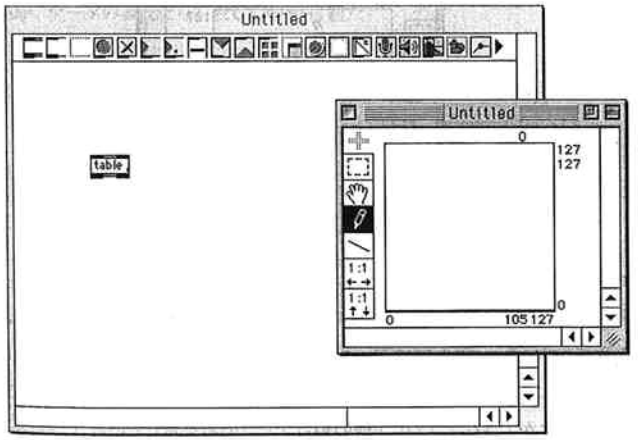
3-13 データの処理

データを処理する場合に、そのデータが数値計算できる場合と数値計算では扱えない場合がある。例えば、銀行預金の利子計算なら数値計算できるだろう。預金する金額と利率などの条件が決まれば、何日先であっても利子がいくらになるかを計算できる。一方、毎日の気温をグラフ化したい場合には、数値計算はできない。何らかの方法で1日ごとの気温を記憶し、その気温データを呼び出してグラフとして描画しなければならない。ここでは、そのようなデータを扱う代表的なオブジェクトを紹介する。

tableオブジェクトによる変換テーブル

tableは、一定範囲の整数に対応する整数の組み合わせを記憶し、呼び出すことができるオブジェクトだ。そのため、tableオブジェクトを変換テーブル(変換表)として用いることになる。パッチ・ウィンドウにtableオブジェクトを作成すると、同時にテーブル・ウィンドウが開かれる。このテーブル・ウィンドウで整数の対応を指定することができる。

■3-13-1　tableオブジェクトとテーブル・ウィンドウ

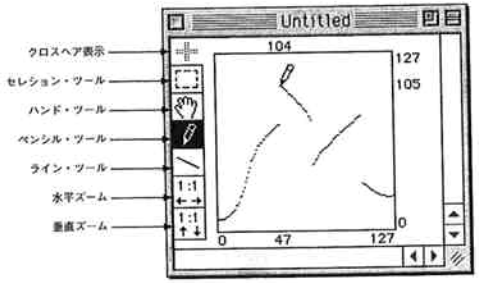


テーブル・ウィンドウでは、水平方向のx軸の値に対する垂直方向のy軸の値を設定する。ペンシル・ツールが選択されていれば、領域内でマウスをドラッグしてグラフが描けるだろう。点描のようにクリックを繰り返してもよい。ライン・ツールが選択されている場合は、直前の位置からクリックした位置を結ぶ直線が描かれる。

正確に数値を設定したい場合は、領域の下側と右側にマウス・ポインターの位置が表示されているので、それを参考にすればよい。上側には設定されたy軸の値が表示される。クロスヘア表示を有効にすれば、マウス・ポインターの位置に合わせて補助線が描かれるので、さらに分かりやすいだろう。水平ズームや垂直ズームの矢印をクリックすれば、表示を拡大または縮小することができる。全領域がウィンドウ内に収まらない場合は、スクロール・バーやハンド・ツールを使って、表示領域を変えればよい。また、セレクション・ツールを使って特定の領域を選択後、Editメニューのコマンドで選択領域のカットやペーストなどが行える。

ちなみに、テーブル・ウィンドウを閉じてもデータは保持される。いったん閉じたテーブル・ウィンドウを再び開くには、tableオブジェクトをダブル・クリックすればよい。パッチがアンロック状態であれば、commandキーを押しながらダブル・クリックする。

■3-13-2　テーブル・ウィンドウの利用



このようなツールを使って、テーブル・ウィンドウでx軸の値に対するy軸の値を設定する。初期状態では、x軸は0から127までの範囲であり、これはデータの数が128個あることを意味している。x軸の値はデータのインデックス番号と考えればよい。同じく初期状態ではy軸は0から127までなので、128個のデータそれぞれが、0か127までの範囲の値を取ることができる。つまり、y軸の値はデータの値を示している。

tableがどのようなデータを扱うかは、インスペクターを使って設定することができる。

■3-13-3　tableオブジェクトのインスペクター

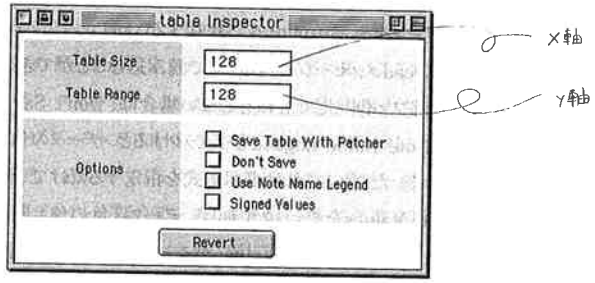


Table Sizeはデータ数のことで、x軸の大きさになる。これが10なら10個のデータを扱うことになり、x軸は0から9まで範囲になる。インデックス番号も0から9までとなる。これに対してTable Rangeはデータの値が取り得る範囲で、y軸の大きさに相当する。これが20なら各データは0から19までの値を取ることになる。この場合、3-13-4の左側のようにデータが表示される。テーブル・ウィンドウでは各データが点で表されるが、これを棒グラフのように考えると分かりやすいいいかもしれない。実際にこのような表示はできないが、データの概念として表したのが右側だ。

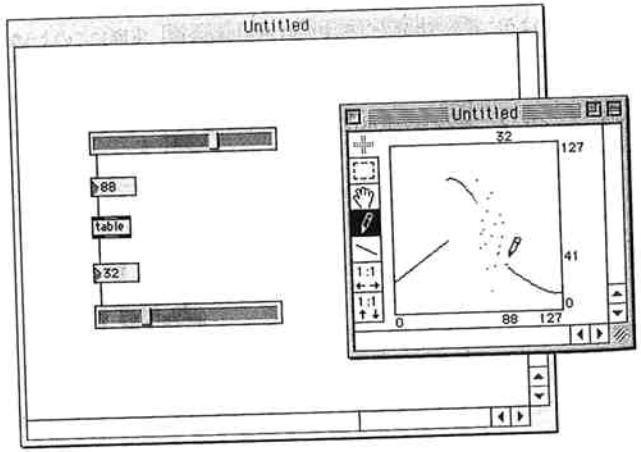
■3-13-4　テーブル・データの実際の表示（左側）と概念図（右側）

05/3/12

Save with Table With Patcherをチェックすると、tableオブジェクトのデータはパッチ・ファイルに保存される。つまり、パッチを閉じてのち、もう1度パッチを開いたときに同じデータが利用できるわけだ。この項目をチェックしていない場合は、パッチを閉じる際にtableオブジェクトのデータを外部のファイルに保存することになる。ファイルとして保存したデータは、tableにreadメッセージを送ることで読み込むことができる。データをパッチ・ファイル内にも外部ファイルとしても保存しない場合は、Don't Saveをチェックすればよい。また、Use Note Name Legendをチェックすると、データの値はC3やF#4といった音名で表示される。ただし、これは表示形式を指定するだけで、値としては整数のままである。Signed Valuesをチェックすれば、データは負の値も取るようになる。例えば、Table Rangeが100でSigned Valuesがチェックされている場合、データの値は−100から99までの範囲となる。

さて、適当なtableオブジェクトのデータを作成できたらそれを利用してみよう。tableは整数を受け取ると、その整数をインデックス番号と見なして、対応するデータの値を出力する。3-13-5のパッチでスライダーを動かせば、tableデータに応じた数値が出力されることが分かるだろう。

■3-13-5　tableオブジェクトの処理

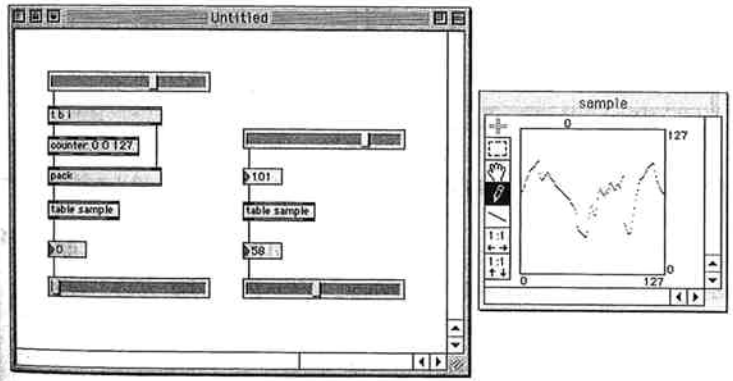


一方、tableオブジェクトにデータを設定するには、2つの整数から成るリストを送ればよい。リストの最初の要素をインデックス番号とし、2番目の要素をy軸の値とするようなデータが設定される。

3-13-6の例の左側のパッチでは、hsliderオブジェクトのスライダーを動かすとtableオブジェクトのデータが設定されるようになっている。ここでは、counterオブジェクトの出力をインデックス番号とし、hsliderオブジェクトの出力をデータの値として扱い、これらをpackオブジェクトでリストにまとめてtableオブジェクトに送っている。

ところで、tableオブジェクトにはsampleというアーギュメントを与えている。これはtableオブジェクトが扱うデータの集まりに対して名前を指定することにある。同じ名前を指定したtableオブジェクトは、同じデータを共有する。したがって、左側のパッチで設定したtableオブジェクトのデータは、右側のパッチで読み出すことができる。

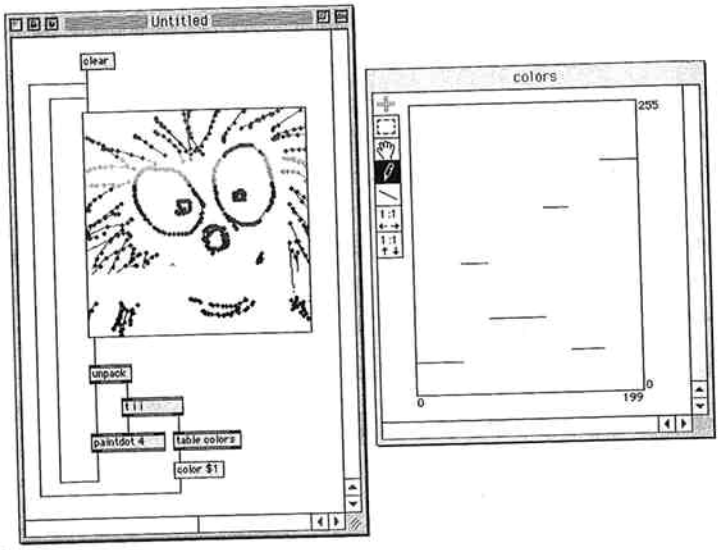
■3-13-6　tableオブジェクトのデータの設定とデータの共有



では、tableオブジェクトを使って色を変えながら描画するパッチを作ってみよう。このパッチでは、lcdオブジェクトをドラッグしているマウスのy座標をインデックス番号としてtableオブジェクトのデータを読み出し、その値を描画色として設定している。テーブル・ウィンドウで、段階的なデータを作った場合は、lcdオブジェクト上でマウスを動かして絵を描くと縦縞のように色が付くことになる。

05/3/12

■3-13-7 tableによって色を変えて描画する処理



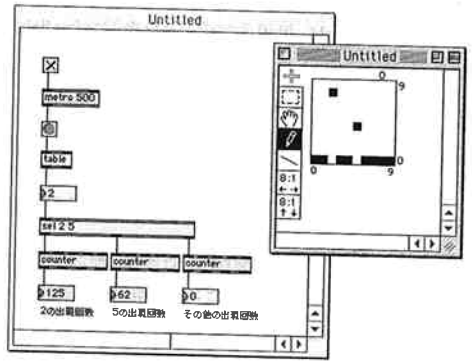
05/3/12

tableオブジェクトによる確率テーブル

tableオブジェクトは、変換テーブルとして用いる以外に確率テーブルとして用いることができる。これは、tableオブジェクトがbangメッセージを受け取ったときに、各データの値を相対的な出現率として、その頻度に応じたインデックス番号を出力するものだ。例えば、Table SizeとTable Rangeがともに10のテーブルで考えてみよう。インデックス番号2のデータの値が8、インデックス番号5のデータの値が4、それ以外のデータの値は0とする。これで、何回かbangメッセージをtableオブジェクトに送ると、tableからは2と5が2：1の割合で出力され、他の数値は出力されない。

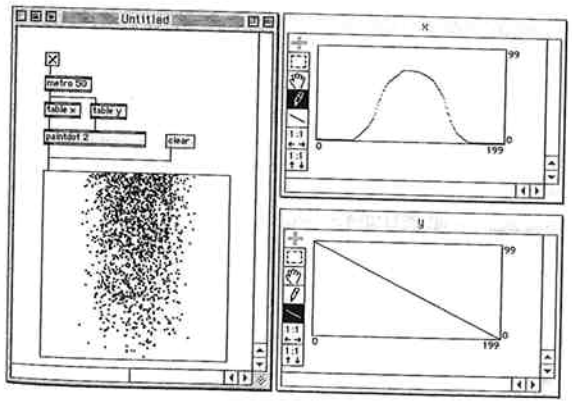
次のパッチでは、metroオブジェクトを使って繰り返してbangメッセージをtableオブジェクトに送り、2と5をsel 2 5で選択し、counterオブジェクトで出現回数を数えている。数回程度では偏りがあるかもしれないが、数十回も繰り返せば、2と5がほぼ2：1の割合で出力されていることが分かるだろう。

■3-13-8 確率テーブルとしてのtableオブジェクトの動作



では、確率テーブルに基づいて座標を決定しドットを描いてみよう。次のパッチではtable xとtable yという2つのtableを用意している。lcdオブジェクトの横幅と縦幅が200なので、いずれもTable Sizeを200とし、Table Rangeを100としている。それぞれのテーブル・ウィンドウで確率テーブルを設定し、toggleオブジェクトをチェックしてドットを描いてみよう。3-13-9の設定では、x座標は中央付近が頻度が高くなり、y座標は上端から下端へと頻度が次第に減少していくのが分かる。

■3-13-9 確率テーブルによる描画



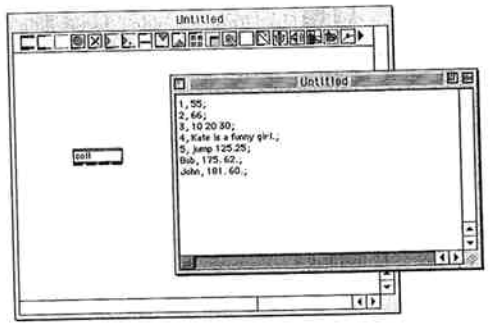
collオブジェクトによるデータ操作

tableオブジェクトは整数しか扱えないが、汎用的にデータを扱うにはcollオブジェクトを利用するのがよいだろう。collは、キーとなる整数またはシンボルに対して、値となる任意のメッセージを組み合わせて、1つのデータとし、任意の数のデータを記憶し、呼び出すことができる。collとはcollection（収集、集合）の略だ。

さて、パッチ・ウィンドウにcollオブジェクトを作成したら、これをcommandキーを押しながらダブル・クリックしてみよう。パッチをロックした状態ならダブル・クリックするだけでよい。これでテキスト・ウィンドウが開くので、テキストとしてキーと値の組み合わせを入力することができる。この書式は、<キー>,<メッセージ> ; となる。複数のデータを持たせる場合は、改行しながら各行に記述すればよい。

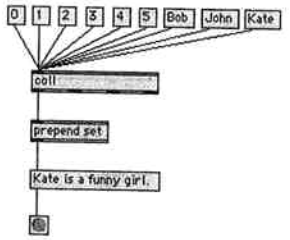
データを作成したなら、クローズ・ボックスをクリックしてテキスト・ウィンドウを閉じる。この際、Save changes to 'Untitled' before closing?とたずねられるので、Saveボタンをクリックして保存する。これは、テキスト・ウィンドウの内容をcollオブジェクトの内部データとして格納するか？といった意味だ。データの形式が正しければ、finished,7 linesといった具合に、Maxウィンドウに格納されたデータ数が表示される。何らかの問題があれば、error : coll : : errors in text files in line 6などのようにエラーが起こった行が表示される。例えば、実数をキーとすることはできない。値を持たずキーだけのデータはエラーにならないが、キーのない値だけのデータはエラーになる。また、同じキーを持つ複数のデータがあっても構わない。

■3-13-11　collオブジェクトのデータ編集



正しくデータが格納されたら、メッセージによってデータを取り出してみよう。このためにはキーになる整数またはシンボルをcollオブジェクトに送ればよい。これでcollオブジェクトの第1アウトレットから、キーに対応する値がメッセージとして出力される。3-13-12の例では、1を送れば55が出力され、Bobを送れば175. 62.が出力されるだろう。キーが一致するデータがなければ、何も出力されない。同じキーを持つ複数のデータがある場合は、最初に見つかったデータの値が出力される。

■3-13-12　collオブジェクトからのデータの取り出し



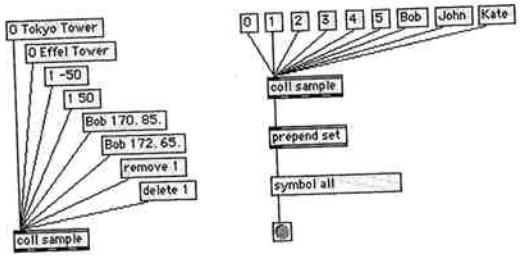
一方、メッセージによってデータを格納することができる。これは、キーと値からなるメッセージをcollオブジェクトに送ることで実現できる。メッセージの最初の要素がキーと見なされ、2番目以降の要素が値となる。キーが同じデータが存在すれば、そのデータは新しい値で置き換えられる。データを格納することを明示するために、最初にstoreというシンボルを付けることも可能だ。

存在しているデータを削除するには、removeまたはdeleteにキーを続けたメッセージを送ればよい。removeの場合は単純にキーが一致するデータが削除されるだけが、deleteの場合はキーが整数であれば、削除されたデータ以降のキーの数値が減少する。そして、clearメッセージを送れば、すべてのデータを削除することができる。

また、collオブジェクトはアーギュメントにシンボルを指定して、collオブジェクトが扱うデータの集まりに名前を付けることができる。次の3-13-13例では、左側のパッチでsampleという名前を指定したcollに対して、データの格納やデータの削除を行うようにしてあり、右側のパッチでもsampleという名前を持つcollオブジェクトからデータを読み出そうとしている。これらの2つのcollオブジェクトは同じ名前をアーギュメントに指定し

ているので、同じデータの集まりを扱うことになる。そのため左側のパッチで格納したデータを、右側のパッチで読み出すことができるのである。

■3-13-13 collオブジェクトのデータ格納と削除



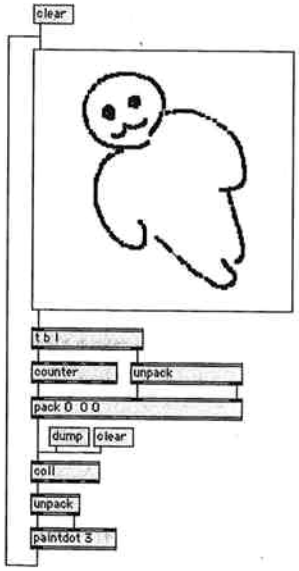
collオブジェクトがインスペクターで設定できる項目はSave Coll with Patcherだけだ。これをチェックしておけば、collに格納されたデータは、パッチ・ファイル内に保存される。この項目をチェックしていないときは、パッチを閉じるとcollオブジェクトの内容は失われる。しかし、パッチ・ファイルを開く際に、collオブジェクトのアーギュメントで指定した名前と同じファイルがあれば、そのファイルの内容が自動的に読み込まれるようになっている。

■3-13-14 collオブジェクトのインスペクター

では、collオブジェクトを用いてマウスを動かした座標を記録するパッチを作ってみよう。3-13-15のパッチでは、lcdオブジェクトの第1インレットから出力される座標のリストをcollに記録する。ただし、counterを使って次第に増加する数値をキーとして用い、その後にx座標とy座標の値を続けた3つの整数から成るリストをcollに送っている。

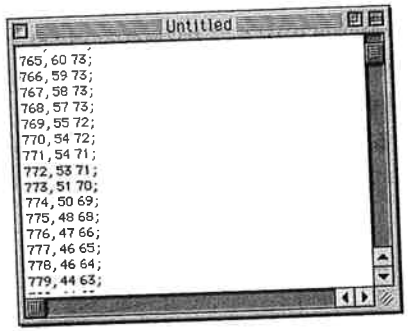
これで、lcdオブジェクト上でマウスをドラッグすれば、その座標が次々とcollオブジェクトに蓄えられていく。次にdumpメッセージをcollオブジェクトに送れば、記録されたすべてのデータの値が第1アウトレットから出力される。これは座標値のリストに他ならないので、unpackオブジェクトでx座標とy座標の値に分けてpaintdotオブジェクトに送り、描画を行う。lcdオブジェクトを消去しても、dumpメッセージをcollオブジェクトに送れば何度でも描き直すことができるだろう。clearメッセージをcollオブジェクトに送れば記録されたデータが消去されるので、新たな記録を行うことになる。

■3-13-15 collを使った座標の記録



ちなみに、座標値を記録したcollオブジェクトの内容をテキスト・ウィンドウに表示すれば3-13-16のようになる。キーとして連続した数値が続き、その値としてマウスの座標が記録されている。そのため、このパッチを発展させれば、特定の範囲のキーを指定して部分的に描画することや、一定のテンポで描画を行うこともできるだろう。

■3-13-16 記録されたcollオブジェクトの内容



なお、メモリーに余裕があれば、collオブジェクトにはかなり大量のデータを格納することができる。とはいえ、テキスト・ウィンドウに表示できるのは32KBまでという制限があるため、32KB以上のデータはテキスト・ウィンドウでは確認や編集ができない。しかし、メッセージによってデータを格納し、取り出すことはできるようになっている。

これ以外にも、collオブジェクトはさまざまな機能を持っており、例えば、データのソート(並び替え)、キーのリナンバー、データのスワップ(入れ替え)、データへのポインターの指定、ポインターが示すデータの出力、などが挙げられる。このように、collオブジェクトは多機能であり、柔軟な活用ができるようになっている。

❷ mtrオブジェクトによるマルチトラック操作

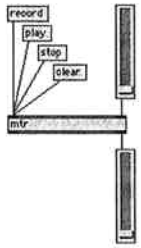
時間的に変化するデータを、時間とともに記録するにはmtrオブジェクトが便利だ。mtrはMulti-track recorderの略で、その名の通り、複数のメッセージを並列的に記録し、再現することができる。

mtrオブジェクトはデフォルトでは内部的に1つのトラックを持ち、2つのインレットと2つのアウトレットを持っている。第1インレットはmtrオブジェクトに対する操作のメッセージを受け取り、第2インレットで記録すべきメッセージを受け取る。記録されたメッセージは第2アウトレットから出力される。

3-13-17のパッチで、recordメッセージを第1インレットに送れば、mtrオブジェクトは記録状態になる。上部のuslider《Vertical Slider》のスライダーを上下に動かせば、内部のトラックに受け取った値が時間と共に記録されていく。

次にplayメッセージを送れば、記録されたメッセージが時間経過に従って再生される。下側のスライダーが先ほどの操作と同じように動くのが分かるだろう。stopメッセージは記録や再生を停止し、clearメッセージは記録した内容を消去する働き持つ。

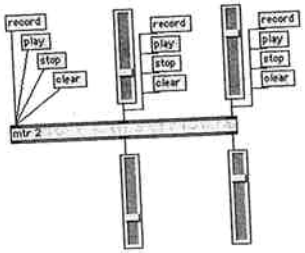
■3-13-17 mtrオブジェクトの動作



mtrオブジェクトはアーギュメントで32までのトラック数を指定する。これで、第2インレットと第2アウトレット以降、指定したトラック数分のインレットとアウトレットが作られ、それぞれのトラックに対応する。第1インレットに送ったrecordやplayなどのメッセージは全トラック同時に操作するが、第2インレット以降にメッセージを送ることで各トラックごとの操作も可能だ。3-13-18の例では、2トラックのmtrオブジェクトを作っている。本来これらのトラックは同時に記録することができるが、マウスでは1つのスライダーしか操作できない。そこで、まず第2インレットにrecordメッセージを送り、左側のスライダーを操作する。しばらく操作をしたら、第2インレットにstopメッセージを送る。次に、第3インレットにrecordメッセージを送って、右側のスライダーをしばらく操作する。そして第3インレットにstopメッセージを送る。これで2トラック分の記録ができたので、第1インレットにplayメッセージを送って、2つのトラックを同時に再生する。

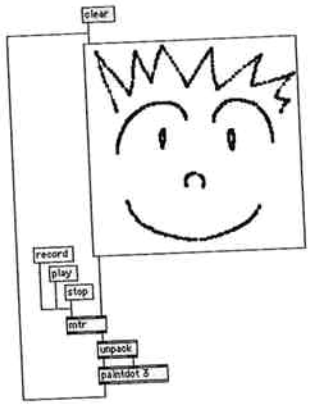
これで下部の2つのスライダーが同時に動く。もちろんトラックごとに再生することもできれば、第1トラックを再生しながらの第2トラックの記録も可能。またここでは整数で記録しているが、mtrオブジェクトはどのような種類のメッセージでも記録/再生可能だ。

■3-13-18 2トラックのmtrの動作



ではmtrオブジェクトを使ってマウスの動きを記録し、再現するパッチを作ってみよう。3-13-19では、lcdオブジェクトの第1アウトレットからはマウス・ドラッグした際の座標がリストとして出力されるので、これをmtrオブジェクトで記録する。再生時はmtrオブジェクトから出力されるリストをunpackオブジェクトでx座標とy座標に分け、paintdotオブジェクトに送り描画を行うようにしてみた。recordメッセージ・ボックスをクリックし、lcdオブジェクト上でマウスをドラッグしてみよう。次にplayメッセージ・ボックスをクリックすれば、先に描いた通りにドットが描かれる。もちろん描く速度も同じように再現される。

■3-13-19 mtrによるマウスの動きの記録と再現



05/3/12

データのファイル入出力

tableやcollなどのオブジェクトは、インスペクターで指定することでそのデータをパッチ・ファイル内に保存したり、データを独立したファイルとして保存し、ファイルからデータを読み込むことができる。ここではcollオブジェクトについてファイルの入出力の概要を説明するが、tableなど他のデータを扱うオブジェクトでも同様の処理が可能である。

まず、collオブジェクトの内容をファイルとして保存するには、collオブジェクトにwriteメッセージを送ればよい。また、collの内容を編集するテキスト・ウィンドウが開いているときに、FileメニューのSaveやSave As...を選んでも構わない。

いずれも、ファイル保存ダイアログが開くので、適切なファイル名を付けて保存する。ファイル名にはスペース(空白文字)を含めない方があとの処理が簡単になる。例えば、data 1ではなくdata1とした方がよい。スペースを含まない名前は、そのままシンボルとして扱えるからだ。

スペースを含む文字列をファイル名として扱いたい場合は、ファイル名を'sample data'のように ' と ' で囲んで表記すればよい。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 [キーを押すことで入力する。

特定のファイル名で保存するには、writeに続けてファイル名を指定したメッセージをcollオブジェクトに送る。ファイル名はシンボルとして指定しなければならない。例えば、write data1というメッセージを送ると、collオブジェクトの内容がdata1というファイル名で保存されることになる。この場合、ファイル保存ダイアログは開かない。

一方、readメッセージを送れば、ファイル選択ダイアログが開くので、選択したファイルからデータを読み込むことができる。readに続けてファイル名を指定したメッセージをcollオブジェクトに送れば、ダイアログが開くことなく、そのファイルの内容がcollオブジェクトに読み込まれる。ただし、ファイル名はシンボルとして指定し、ファイルはMaxのサーチ・パス内に存在しなければならない。

例えば、read data1というメッセージをcollオブジェクトに送れば、data1というファイルの内容が読み込まれる。collオブジェクトにアーギュメントとしてシンボルを指定している場合は、パッチを開くと同時にアーギュメントと同じ名前を持つファイルの内容を自動的に読み込むことになる。このファイルもMaxのサーチ・パス内から探され、仮にア

ーギュメントをcoll data1としていれば、data1という名前のファイルの内容が読み込まれることになる。

なお、collオブジェクトのインスペクターでSave Coll with Patcherをチェックしていれば、collの内容がパッチ・ファイル内に保存される。また、collのアーギュメントと同じ名前のファイルが存在している場合は、そのファイルの内容が読み込まれるが、Save Coll with Patcherの指定が優先されるので、この項目をチェックしている場合は、ファイルの内容は読み込まれない。

05/3/12

3-14 Maxとパッチのコントロール

Maxでは、Max自体の動作や、パッチ・ウィンドウの動作や見かけなどを、メッセージによってコントロールすることができる。また、パッチ自体の作成や変更もメッセージによってできる。このことを利用すれば、高度な処理が可能になるが、それだけでなく、パッチの動作条件を整えるためや、使いやすいパッチを作るために活用できるだろう。

Maxのコントロール

Maxアプリケーションにメッセージを送り、Maxをコントロールすることができる。このためには、メッセージ・ボックスに ;maxで始まるメッセージを作り、これをクリックするか、bngメッセージを送ればよい。それほど多くのことができるわけではないが、一般に配布するパッチを作成する場合に、Maxがどのようにパッチを動作させるかという動作環境を設定するために活用できるだろう。例えば、;max preempt 1ならオーバードライブをonにし、;max preempt 0ならオーバードライブをoffにすることができる。これで、ユーザーの設定に関わらず、パッチによってオーバードライブを望む状態に設定することができる。オーバードライブは、表示よりも内部処理を優先するモードで、OptionsメニューのOverdriveをチェックすれば、オーバードライブがonの状態になる。

また、;max hidemenubarというメッセージならメニュー・バーが消去され、;max showmenubarでメニュー・バーが表示される;max quitならMaxアプリケーションを終了させることができる。ちなみに、一行でメッセージを入力しても、次にパッチを開いたときには ; とmaxとの間が改行されて表示されるようになっている。

■3-14-1 Maxへのメッセージの例