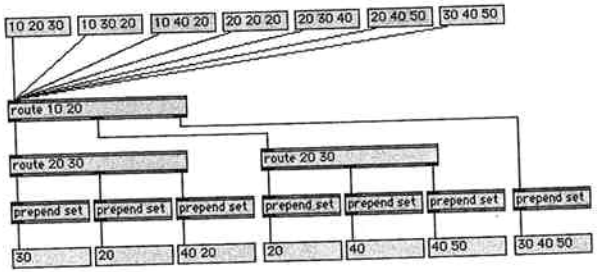
メッセージが2つまたは3つ以上の要素から成っている場合は、routeオブジェクトの1番目以降のアウトレットに、さらにrouteオブジェクトをつなぐことが考えられる。この場合は、最初のrouteオブジェクトのアーギュメントに一致した場合に、2番目以降の要素からなるメッセージが出力される。そして出力されたメッセージは、次のrouteオブジェクトでアーギュメントに一致するかを調べて、該当するアウトレットから出力される。つまり、最初のメッセージでの2番目の要素を調べることになる。このようにして、複数の要素からなるメッセージを要素ごとに細分化して調べていくことができる。

3-11-19の例では、10 20 30というメッセージを送ると、route 10 20では1番目のアーギュメントに一致するので、第1アウトレットから20 30が出力される。この20 30は、次のroute 20 30では最初のアーギュメントに一致するので、第1アウトレットから30が出力され、一番左のメッセージ・ボックスに30が表示される。

他のメッセージについても、どのように処理されるか考えてみよう。

■3-11-19 routeオブジェクトを連ねた処理



3-12 時間的な処理

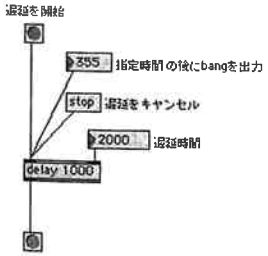
Maxは音楽に関連する機能が豊富なこともあり、時間を扱うオブジェクトが数多く用意されている。ここでは時間処理のための基本的なオブジェクトを紹介しよう。具体的には、時間的な遅延を行う処理、時間的にメッセージを制限する処理、時間を計測する処理、時間的な動作を行う処理、そして日付や時刻を得る処理について説明する。

delayオブジェクトによる時間遅延

delayオブジェクトは、第1インレットにbangメッセージを受け取ると、アーギュメントで指定した時間だけ遅らせてbangメッセージを出力する。第2インレットに数値を入力すれば、遅延時間を設定することができる。また、第1インレットに数値を送ると遅延時間を変更し、その時間の経過後にbangメッセージを出力する。ここでの時間はすべてmsec単位だ。なお、第1インレットにstopメッセージを送れば、遅延中の処理を取り消すので、bangメッセージは出力されない。delayはdelと省略形を用いることができる。

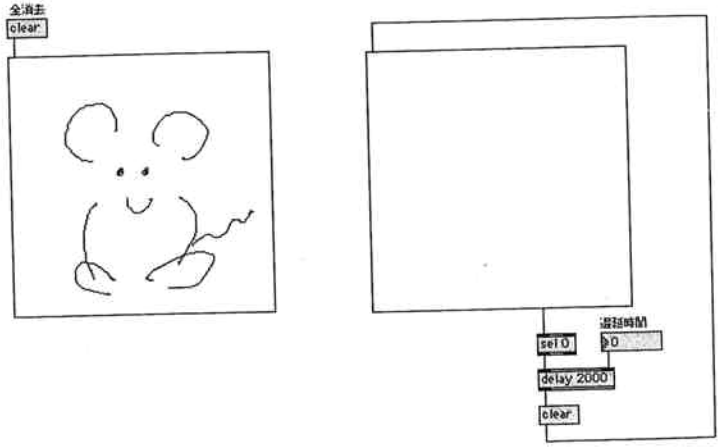
次のパッチで、上部のbuttonオブジェクトをクリックすると、一定時間ののちに下のbuttonオブジェクトが点滅するのが分かるだろう。delayが時間を遅らせてbangメッセージを出力しているからだ。delayによって遅延できるbangメッセージは1つに限られる。遅延中に新しいbangメッセージを受け取ると、遅延中であった処理は中断され、新たな遅延が始まる。したがって、buttonオブジェクトを素早く何度かクリックしても、最後にクリックしてから一定時間後に1度だけ下のbuttonオブジェクトが点滅する。

■3-12-1 delayオブジェクトによる時間遅延



では、delayを用いたパッチの例として、一定時間後に自動的に画像が消去される落書き板を作ろう。まず、lcdオブジェクトへメッセージを送ることによって描画できるが、それ以外にもlcdオブジェクト内でマウスをドラッグすれば、線画を描くことができる。しかも、マウスの情報がlcdオブジェクトから出力されているので、これを利用して自動消去機能を作ればよい。

■3-12-2　マウス・ドラッグによる手書き風の描画　■3-12-3　描画して2秒後に消去するパッチ

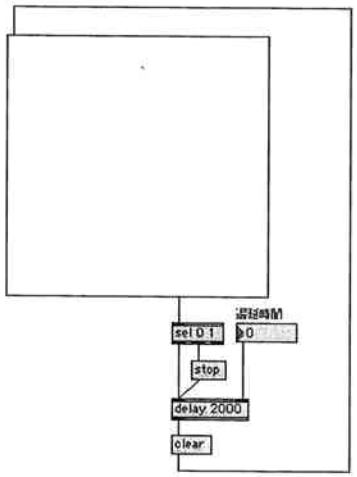


lcdオブジェクトの第3アウトレットからは、マウス・ボタンを離した時点で0が出力されるので、select 0によって0を受け取ったときにbangメッセージを出力することができる。このbangメッセージによってclearメッセージをlcdオブジェクトに送れば、画像が消去される。ただし、消去するまでに多少の時間間隔を置きたいので、delay 2000によって2秒(2,000msec)間のずれを設けている。遅延時間はナンバー・ボックスで変更してもよい。

しかし、これだけでは、何か描いたあとにさらに描き加えようとしても、書いている側から次々と指定時間後に消去されてしまう。そこで、続けて描画している間は自動的に消去しないように改良しよう。

これにはマウス・ボタンを押したときにlcdオブジェクトの第3アウトレットから1が出力されることを利用する。具体的にはselectのアーギュメントに1を追加し、1を受け取った際にstopメッセージをdelayオブジェクトに送ればよい。これで、マウス・ボタンを離しても遅延時間が経過しない間にマウス・ボタンを押して描画を始めれば、遅延処理は取り消される。つまりclearメッセージがlcdオブジェクトに送られないので、消去されない。lcdオブジェクトが消去されるのは、描き始めないまま時間が経過した場合になる。

■3-12-4　連続的に描画できるように改良



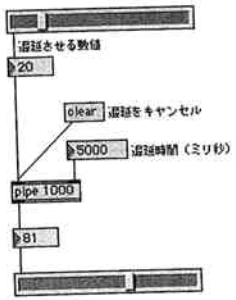
❷ pipeオブジェクトによる整数の時間遅延

delayオブジェクトによって遅延できるのはbangメッセージだけであり、数値を時間的に遅らせて出力したい場合は、pipeオブジェクトを用いなければならない。基本的なpipeオブジェクトの使い方はdelayオブジェクトと似ている。pipeオブジェクトは数値を受け取ると、アーギュメントまたは第2インレットで指定したmsec単位の時間だけ遅らせて、その数値を出力する。delayオブジェクトでの遅延取り消しはstopメッセージだったが、pipeの遅延処理を取り消すにはclearメッセージを用いる。

また、delayオブジェクトで遅延できるbangメッセージは1つに限られていたが、pipeオブジェクトではそのような制限はなく、いくつもの数値を時系列に沿って遅延させることができる。

では3-12-5のようなパッチで上部のスライダーを動かしてしてみよう。hsliderオブジェクトから出力された整数は、pipeオブジェクトによって時間的に遅れて出力され、下のスライダーの位置を変える。この際、上のスライダーの動きが一定時間後に下のスライダーで再現されるように見えるだろう。これは、hsliderオブジェクトから出力されるすべての整数を、pipeオブジェクトが時系列に従って並列的に遅延させているからだ。

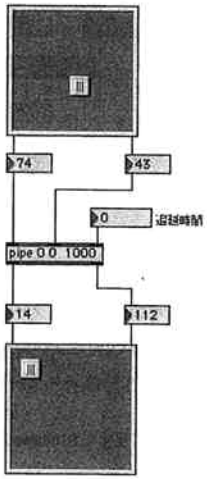
■3-12-5 pipeオブジェクトによる整数の遅延



また、pipeオブジェクトは同時に複数の数値を遅延させることもできる。この場合、扱う個数のアーギュメントと遅延時間とを指定する。例えば、2種類の整数を遅延させたい場合は、pipe 0 0 1000のようにアーギュメントを与える。3-12-6のパッチでは、pictslider《Picture-based slider》オブジェクトのX座標とY座標という2つの整数をpipeオブジェクトによって遅延させている。pictsliderオブジェクトはオブジェクト・パレットの最後から11番目にある。このパッチでも、上のスライダーをドラッグすると、その動きが一定時間後に下のpictsliderオブジェクトで再現される。

ただし、pipeオブジェクトはright-to-left orderに従って、第1インレットに数値を受け

■3-12-6 pipeオブジェクトによる2つの整数の遅延

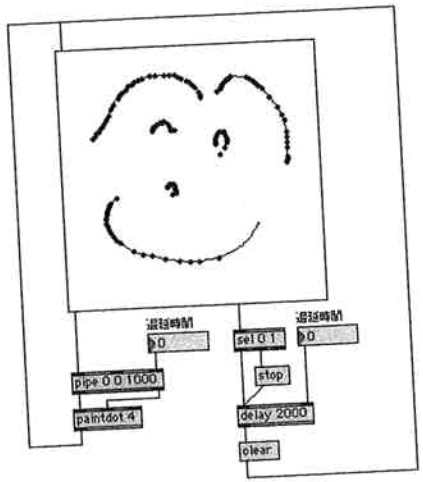


取ったときに遅延を行う。そのため第2インレットに数値を送っただけで、その数値が遅延されて第2アウトレットから出力されるわけではない。なお、アーギュメントに実数を与えると、実数として遅延を行うことができる。

では、pipeの応用として、マウスでlcdオブジェクトに線を描くと、しばらくあとにその動きを追いかけてドットを描くパッチを作ってみよう。

まず、lcdオブジェクト上をマウスでドラッグしているときは、その座標が2つの整数から成るリストとして出力される。このリストをpipe 0 0 1000オブジェクトに入力する。pipeはリストを受け取った場合は、その要素を第1インレットから順に振り分けて処理するので、リストのままpipeオブジェクトに入力すればよい。pipeオブジェクトは指定した時間だけ遅らせて2つの整数を出力するので、これらをpaintdotオブジェクトに送り、ドットを描くメッセージをlcdオブジェクトに送る。これで、マウスで描いた線の後ろを、少し遅れてドットが描かれていく。

■3-12-7　マウスの描画を追いかけてドットを描く

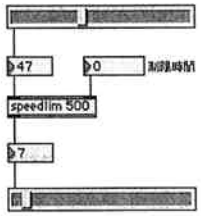


speedlimオブジェクトによる速度制限

先のパッチではマウスをゆっくり動かすと線がドットで覆い隠されてしまうだろう。これはlcdオブジェクトから頻繁にマウス座標を示すメッセージが出力されているからだ。そのため、まばらにドットを描きたい場合には、単位時間当たりのメッセージ量を減らさなければならない。これには、speedlimというオブジェクトが利用できる。

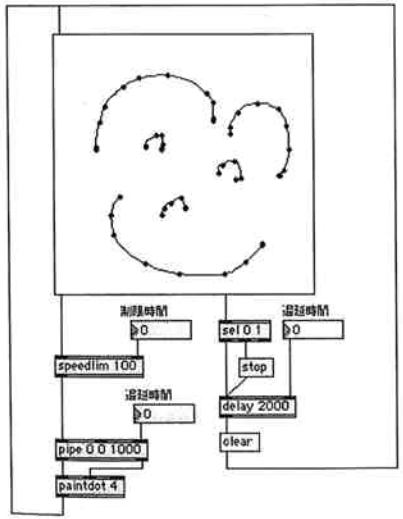
speedlimは、アーギュメントまたは第2インレットに入力した数値をmsec単位の時間として、その時間ごとに受け取ったメッセージを1つだけ出力する。つまり、大量のメッセージを一定時間間隔で間引く働きをする。メッセージはどのような種類であっても構わない。次のようなパッチで上のhsliderオブジェクトのスライダーを素早くドラッグしてみよう。hsliderオブジェクトからは多くのメッセージが出力されるが、speedlimオブジェクトによって、一定時間ごとにしかメッセージが出力されない。したがって、下のスライダーは不連続に動くはずである。

■3-12-8　speedlimオブジェクトによってメッセージを間引く



では、まばらにドットが描画されるようにspeedlimオブジェクトを使って先の3-12-7のパッチを改良してみることにする。この場合、第1インレットからの出力をspeedlimオブジェクトにつなぐ方がよいだろう。pipeやpaintdotオブジェクトの後ろにspeedlimオブジェクトをつないでも構わないが、最初にメッセージを間引いてから続く処理を行う方が、コンピューターへの負担が少なくて済むからだ。

■3-12-9　speedlimオブジェクトによってドットを間引いて描画

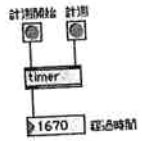


timerオブジェクトによる時間計測

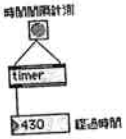
時間の経過を知るためにはtimerオブジェクトを用いるのが簡単だ。timerはタイマーを内蔵したオブジェクトと考えることができ、ちょうどストップ・ウォッチのように利用することができる。timerオブジェクトは、第1インレットにbangメッセージを受け取ると、内部のタイマーを0にして時間計測を開始する。次に第2インレットにbangメッセージを受け取ると、タイマーが示す経過時間をmsec単位の整数として出力する。ストップ・ウォッチに例えるなら、第1インレットはリセット・ボタンとスタート・ボタンを兼ねており、第2インレットはラップ・ボタンに相当する。

3-12-10のパッチで、左側のbuttonオブジェクトをクリックすれば、タイマーが0から動き始める。そして、右側のbuttonオブジェクトをクリックすれば、それまでに経過した時間がナンバー・ボックスに表示される。何度か右側のbuttonオブジェクトをクリックすれば、いずれも左側のbuttonオブジェクトをクリックした瞬間からの経過時間なので、次第に大きな数値が表示されていくだろう。タイマーを0に戻して時間計測を行うには、もう1度左側のbuttonオブジェクトをクリックすればよい。

■3-12-10　timerオブジェクトによる時間計測



■3-12-11　時間間隔の計測



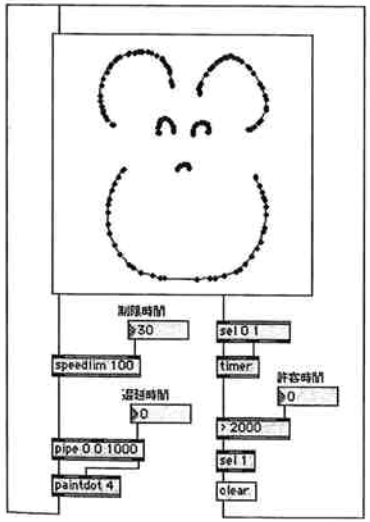
何らかの出来事の時間間隔を計るには、第2インレットにbangメッセージを送ると同時に第1インレットにbangメッセージを送ることを繰り返せばよい。

3-12-11のパッチで、buttonオブジェクトを適当な間隔でクリックすれば、その時間間隔がナンバー・ボックスに表示される。buttonオブジェクトのアウトレットから2つのパッチ・コードが出ているが、right-to-left orderのルールによって、先に第2インレットにbangメッセージが送られて、その後に第1インレットにbangメッセージが送られる。つまり、第2インレットにbangメッセージが送られて経過時間を出力すると同時に、第1インレットにbangメッセージが送られるので、タイマーを0に戻すことになる。このようにして、buttonオブジェクトをクリックする時間間隔が表示されるわけだ。

それでは、timerオブジェクトを利用して描画パッチを改良してみよう。これまでは一定時間何も描かないでいると、自動的に消去していたので、折角描いた絵を長く残すことができなかった。そこで、描き終えてから次に描き始めるまでの時間間隔が指定時間以上であれば、画像を消去するように変更する。このようにすれば、連続的に描画できるとともに、描いた画像はそのまま残るはずだ。そして、一定時間経過したあとに描こうとすると、自動的に画像を消去して描き始めることができる。

このためには、まず、lcdオブジェクトの第3アウトレットの出力をsel 0 1によって判断する。描き終えたときは0が出力されるので、selectオブジェクトの第1アウトレットからbangメッセージが出力され、これをtimerオブジェクトの第1インレットに入力する。描き始めたときは1が出力され、selectの第2アウトレットから出力されるbangメッセージを、timerの第2インレットに入力する。これで、描き終えてから次に描き始めるまでの

■3-12-12　timerオブジェクトを使った自動的な画像消去



経過時間がtimerオブジェクトから出力される。

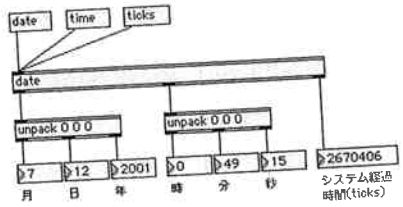
この経過時間を>オブジェクトによって指定時間以上であるかどうか判断する。指定時間以上であれば1が出力されるので、これを受け取ったsel 1オブジェクトがbangメッセージを出力し、clearメッセージがlcdオブジェクトに送られて画像が消去されるという流れだ。

dateオブジェクトによる時刻の取得

timerは経過時間という相対的な時間を得るオブジェクトだが、日付や時刻などの絶対的な時間を得るにはdateオブジェクトが用意されている。特定の時刻に何か動作を起こすようなパッチを作成したいときに利用すればよい。

dateオブジェクトはdate、time、ticksという3つのメッセージを受け取る。dateメッセージを受け取った場合は、第1アウトレットから日付を出力。これは3つの整数から成るリストで、順に月、日、西暦年を表している。timeメッセージを受け取ったdateオブジェクトは、第2アウトレットから現在時刻(これは時、分、秒を表す3つの整数から成るリスト)を出力する。これは24時間制だ。また、ticksメッセージを受け取ると、第3アウトレットからシステム経過時間をticks単位で出力する。システム経過時間とは、コンピューターを起動してから現在に到るまでに経過した時間のことだ。ticksは1/60秒単位なので、第3アウトレットからの出力を60で割った数値が秒数となる。

■3-12-13 dateオブジェクトによる日付や時間の取得



ちなみに、dateオブジェクトはコンピューターに内蔵された時計を参照するので、正しい日付や時刻を得るにためは、コンピューターの内蔵時計を正しく合わせておかなければならない。MacOSの場合は、コントロールパネルの"日付 & 時刻"を使って、内蔵時計を設定する。

■3-12-14 日付 & 時刻コントロールパネル

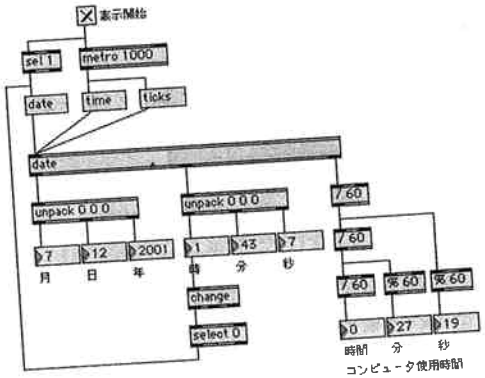
では3-12-13のパッチを発展させて、自動的に時刻を表示するデジタル時計を作ってみよう。時計は1秒ごとに表示を変えればよいので、metro 1000オブジェクトによって1秒ごとにbangメッセージを出力し、これによってtimeメッセージをdateオブジェクトに送ればよい。これでtoggleオブジェクトをチェックすれば、metroオブジェクトが動きだし、1秒ごとに時刻が自動的に表示される。また、同じタイミングでticksメッセージをdateオブジェクトに送っているので、システム経過時間も表示される。ただし、ticks単位では分かりにくいので、時間、分、秒に変換してある。

日付についても毎秒ごとに改めて表示しても構わないが、実際に表示が変わるのは午前0時になったときだけなので、あまりスマートではない。そこで、まず、toggleオブジェクトがチェックされたときに、select 1オブジェクトによってbangメッセージを出力し、dateメッセージをdateオブジェクトに送って日付を表示する。次に日付を更新するのは、午前0時なので、時刻の時間をselect 0オブジェクトで判断して、dateメッセージをdateオブジェクトに送ればよい。

しかし、これでは午前1時になるまでselectオブジェクトはbangメッセージを出力し続ける。そこで、selectオブジェクトの前にchangeオブジェクトを入れる。changeオブジェクトは、入力される数値が変化したときだけ、その数値を出力するというもので、同じ数字が入力された際には何も出力しない。したがって、changeオブジェクトには毎秒ごとに時間の数値が入力されるが、出力するのは時間が変わったときだけである。

これで、毎時0分0秒のときにselectオブジェクトが時間を判断するようになり。午前0時0分0秒の時点でdateメッセージが送られ、日付表示が更新される。

■3-12-15　dateオブジェクトによるデジタル時計



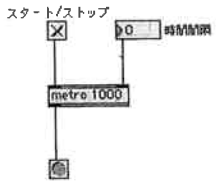
ここまでは、特に表示上の工夫をしていないが、ナンバー・ボックスのフォントのサイズや標示オプションの設定を変えて、より見やすくしてみよう。

また、日本での日付表記に合わせて、年月日の順序にナンバー・ボックスを置き換えてみるのもいいだろう。ナンバー・ボックスとコメント以外のオブジェクトやパッチを隠し、表示するオブジェクトを体裁よく配置したいところだ。このようなレイアウトをうまく行うには、表示に関係しないオブジェクトをサブ・パッチにまとめてしまうことも考えられる。さらに、loadbangオブジェクトを使って、パッチを開くと同時に時計が動くようにするというアイディアもある。

metroオブジェクトによる周期的動作

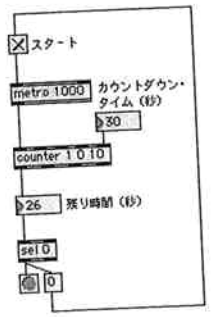
すでに繰り返し処理として説明したが、metroオブジェクトは一定の時間間隔で周期的にbangメッセージを出力する。第1インレットに0以外の数値を受け取れば動作を開始し、0なら動作を止める。アーギュメントと第2インレットに受け取る数値は、metroオブジェクトが動作する時間間隔をmsec単位で指定する。時間間隔は実数でも指定できるので、小数点以下も有効だ。

■3-12-16　metroオブジェクトの動作

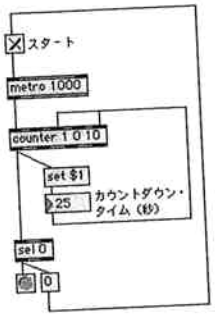


これまではmetroオブジェクトを使って繰り返して描画を行う例を紹介したので、ここではmetroオブジェクトによるカウントダウン・タイマーを作ることにしよう。これは、例えばタイマーを10秒と設定し、スタートさせると、10、9、8、7……といった具合に1秒ごとに数値が減っていき、0になるとbuttonオブジェクトが点灯するという仕掛けだ。具体的には、metroとcounterオブジェクトを用いてパッチを作る。ポイントはcounter 1 0 10のようにcounterの1番目のアーギュメントを1にすることだ。これでcounterオブジェクトはbangメッセージを受け取る度に内部のカウンターを減少させることができる。あとは10、9、8、7……のようにカウンターが変化し、出力される値をナンバー・ボックスで表示させればよい。カウントダウンする秒数の設定は、ナンバー・ボックスで行い、これをcounterオブジェクトの第5インレットにつなぐ。counterオブジェクトの第5インレットはカウンターの最大値の指定なので、ナンバー・ボックスで設定した数値からカウントダウンすることになる。

■3-12-17 カウントダウン・タイマー



■3-12-18 改良したカウントダウン・タイマー



カウントダウンが終わったときにcounterオブジェクトが出力する値が0なので、これをsel 0によって判断し、buttonオブジェクトを点灯させ、metroオブジェクトの動作を止めるために0をtoggleオブジェクトに送ればカウントダウン・タイマーとして動作する。

このカウントダウン・タイマーをさらに改良したのが3-12-18だ。ここでは、1つのナンバー・ボックスだけで、カウントダウンする秒数の設定とカウントダウン中の秒数の表示を行っている。ナンバー・ボックスからの出力をcounterオブジェクトに送ってカウントダウンする秒数を設定している点は3-12-17のパッチと同じだが、counterオブジェクトからの出力を単純にナンバー・ボックスにつなぐと、カウントダウン中の秒数によってcounterオブジェクトの最大値が変更されてしまう。そこで、counterオブジェクトからの出力はset $1によってナンバー・ボックスに表示している。

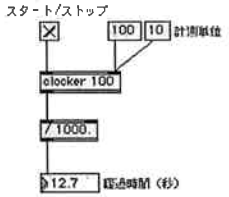
setメッセージはナンバー・ボックスの表示を変えるが、ナンバー・ボックスからの出力を行わない。したがって、カウントダウン中の秒数がcounterオブジェクトに影響を与えることはなく、ユーザーがナンバー・ボックスを操作した場合だけ、counterオブジェクトの最大値を設定することになる。なお、ナンバー・ボックスの出力はcounterオブジェクトの第3インレットにもつなぎ、counterオブジェクトの内部カウンターを設定している。このようにすれば、カウントダウン中に最大値を変更した場合は、その数値からカウントダウンが行われるようになる。

clockerオブジェクトによる経過時間出力

metroに似たオブジェクトとしてclockerがある。clockerオブジェクトはアーギュメントや第2インレットで受け取る数値をmsec単位として周期的な動作を行う。第1インレットに0以外の数値を受け取れば動作を開始し、0なら動作を止める。ここまではmetroオブジェクトと同じだが、metroが一定の時間間隔でbangメッセージを出力するのに対して、clockerオブジェクトは経過時間をmsec単位で出力する点が異なる。

3-12-19の例は、clocker 100としているので、100msec、すなわち0.1秒ごとにclockerオブジェクトが動作するというパッチである。toggleオブジェクトをチェックした瞬間に0が出力され、以後0.1秒経過するごとに、100、200、300……といった具合に経過時間が出力される。

■3-12-19 clockerオブジェクトの動作 ■3-12-20 clockerオブジェクトによるストップ・ウォッチ



それでは、clockerオブジェクトを使ってストップ・ウォッチを作ってみよう。3-12-20のパッチでは、toggleオブジェクトをチェックするとclocker 100によって100msecごとに経過時間が出力される。この出力はmsec単位なので、/ 1000.によって秒単位に変換して、フロート・ナンバー・ボックスに表示する。ここで除算や表示を実数として行っていることに注意しよう。実数として扱わなければ、0.1秒間隔で処理しているにもかかわらず、1秒以下の小数部が失われてしまう。

また、上部のメッセージ・ボックスをクリックすることで、ストップ・ウォッチの計測単位を変えることができる。10というメッセージをclockerオブジェクトの第2インレットに送れば、clockerオブジェクトは10msec間隔で動作するようになり、0、10、20、30……のように経過時間を出力する。したがって、フロート・ナンバー・ボックスには、0.、0.01、0.02、0.03……のように0.01秒単位で経過時間が表示される。