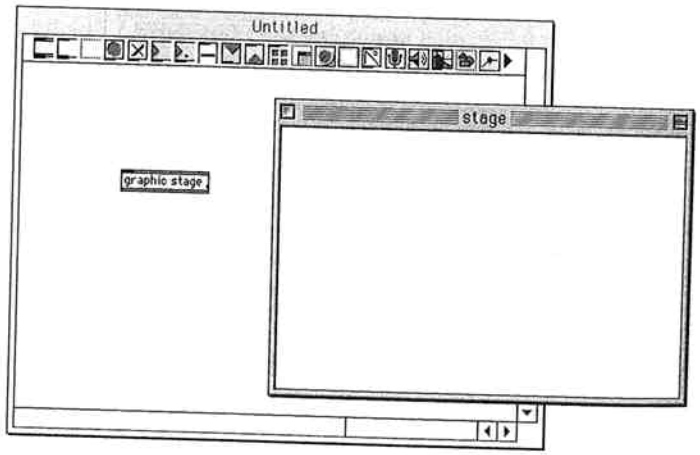
6-4 スプライト・アニメーション

Maxはグラフィック・ウィンドウを開き、そこに幾何学的な図形やビットマップ画像を表示することができる。これらの図形や画像は独立したオブジェクトであり、それぞれを個別に動かし、大きさや色などを変えることができる。これは、他のアプリケーションではFlashやDirectorなどのスプライト・アニメーションに相当すると考えてよいだろう。

graphicオブジェクトによるグラフィック・ウィンドウの作成

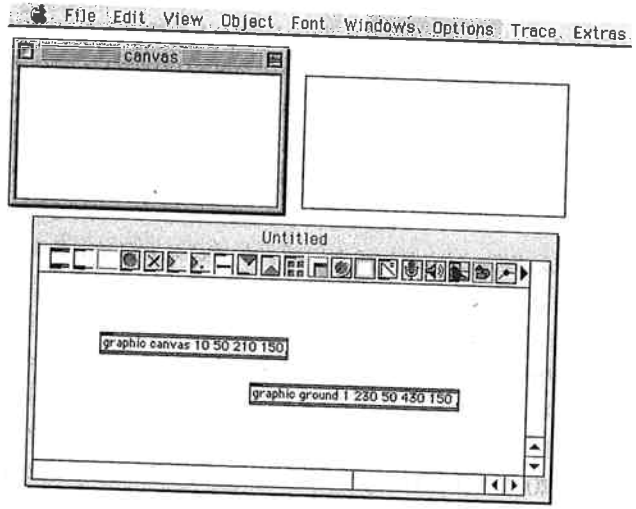
幾何学図形などを表示するグラフィック・ウィンドウを開くには、graphicオブジェクトを用いる。graphicオブジェクトには、アーギュメントとしてシンボルを指定し、これがグラフィック・ウィンドウの名前となり、そのタイトル・バーに表示される。

■6-4-1 graphicオブジェクトによるグラフィック・ウィンドウの作成



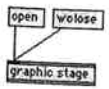
graphicオブジェクトのアーギュメントには、シンボルに続く4つの整数によって、グラフィック・ウィンドウの位置と大きさを指定することができる。これらの整数は、スクリーンの左上隅を原点として、グラフィック・ウィンドウの左端座標、上端座標、右端座標、下端座標を示す。また、アーギュメントとしてシンボルに続いて5つの整数を指定する場合、最初の整数が0以外であれば、グラフィック・ウィンドウはタイトル・バーを持たないことになる。2つ目以降の4つの整数は、グラフィック・ウィンドウの表示座標を表す。

■6-4-2 グラフィック・ウィンドウの位置と大きさ、タイトル・バーの指定



graphicオブジェクトにwcloseメッセージを送ると、グラフィック・ウィンドウを閉じることができる。また、openメッセージなら、グラフィック・ウィンドウを開くことになる。graphicオブジェクト自体をダブル・クリックしてもグラフィック・ウィンドウが開く。

■6-4-3 グラフィック・ウィンドウの開閉メッセージ



graphicオブジェクトは、グラフィック・ウィンドウを作成し、管理するだけの機能を担っている。したがって、グラフィック・ウィンドウに描画するには、次に説明するovalやpictなどのオブジェクトを用いることになる。

frameオブジェクトなどによる幾何学図形の表示

グラフィック・ウィンドウに幾何学的な図形を描くために、frame、rect、ring、ovalの4つのオブジェクトが用意されている。それぞれ、枠線だけの矩形(長方形、正方形)、塗りつぶした矩形、枠線だけの円(楕円)、塗りつぶした円を表示する。これらは形状が違うが、アーギュメントや受け取るメッセージは共通している。ここでは、rectについて説明するが、他のオブジェクトも同じように利用できる。

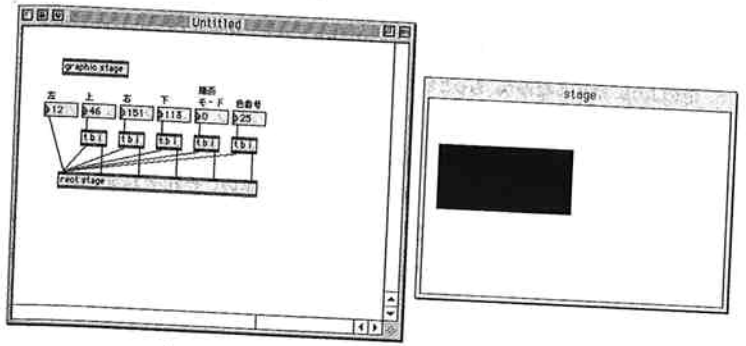
まず、rectのアーギュメントで、矩形を表示するグラフィック・ウィンドウをシンボルによって指定する。当然のことながら、そのグラフィック・ウィンドウが存在している必要がある。例えば、stageという名前のグラフィック・ウィンドウに表示したい場合、rect stageのようにアーギュメントを指定してオブジェクトを作成すればよい。

次にrectのインレットに数値を送り、矩形の座標を指定すれば、それに基づいた表示が行われる。rectの第1インレットから第4インレットによって、順に矩形の左端座標、上端座標、右端座標、下端座標を指定する。これらの座標は、グラフィック・ウィンドウの左上隅を原点とするピクセル単位の座標系だ。ただし、right-to-left orderに従って、第1インレットに数値を受け取ったときに、実際の描画が行われることになる。また、第1インレットにbangを送れば、それまでに受け取った数値によって描画が行われる。

rectの第5インレットは描画モードの指定だが、単独の図形だけでは効果が分かりにくいので、のちほど説明する。そして、rectの第6インレットは、矩形を描画する色の指定で、Maxのカラー・パレットのインデックス番号として0から255までの数値を用いる。この色指定は、次に矩形が描かれるときに用いられるので、直ちに色を変えたい場合は、第1インレットにbangメッセージを送る必要がある。

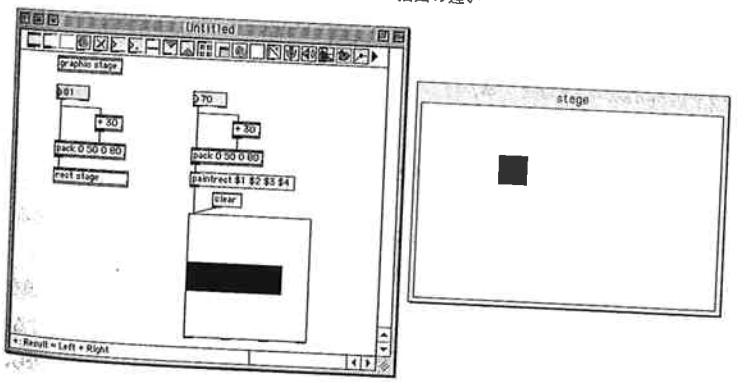
それでは、次のようなパッチで、rectオブジェクトを操作してみよう。ここでは、第2インレット以降に数値を送るときに、triggerオブジェクトによって第1インレットにbangメッセージを送っている。したがって、いずれのナンバー・ボックスを操作しても、その変化がグラフィック・ウィンドウに反映されることになる。

■6-4-4 rectオブジェクトによる矩形描画



ここで重要なのは、rectオブジェクトによって描かれる矩形は、独立した位置と大きさと色を持ったスプライトであることだ。そのため、座標や色を変更するとスプライトが変化し、描き直される。したがって、6-4-5のパッチでナンバー・ボックスをドラッグすれば、30ピクセル正方形が左右に移動するだろう。ちなみに、rectオブジェクトにリストを送れば、第1インレットから順に各要素を送ったことになる。ここでは、4つの数値からなるリストをrectに送っているので、矩形の座標を設定しているわけだ。

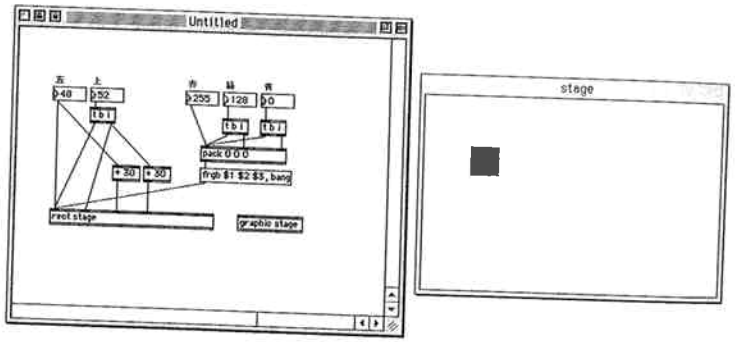
■6-4-5 グラフィック・ウィンドウでの描画とlcdでの描画の違い



これに対して、lcdオブジェクトはビットマップでの描画なので、複数のrectメッセージを送れば、次々と重なって描かれる。6-4-5のパッチの右側では、同じように30ピクセルの正方形をlcdに描いているが、ナンバー・ボックスをドラッグすれば、正方形が重なって、帯のように見えるだろう。もっとも、lcdオブジェクトでもスプライトが使えるので、それはこの限りではない。

なお、フルカラーで色を指定したい場合は、第1インレットにfrgbメッセージを送ればよい。frgbメッセージは3つの整数のアーギュメントを伴い、それぞれ赤、緑、青のRGB値として、0から255までの範囲をとる。この色指定も、次に矩形が描かれるときに用いられるので、直ちに色を変えたい場合は、第1インレットにbangメッセージを送る必要がある。

■6-4-6　フルカラーでの色の指定



cf 映像オブジェクトの表示はできるか？ 05/1/5　> cf P.707　05/1/5 実行せず

pictオブジェクトによる画像の表示

グラフィック・ウィンドウには、pictオブジェクトを用いて画像を表示することもできる。この画像もスプライトとして扱われるので、移動することや、重ね合わせが可能だ。

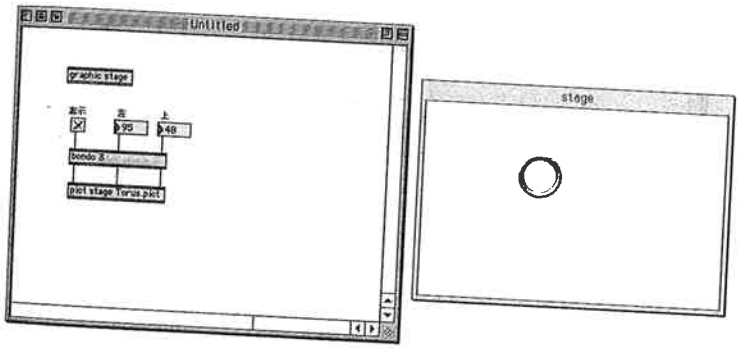
pictオブジェクトは、アーギュメントに画像を表示するグラフィック・ウィンドウの名前を指定する。そして、2番目のアーギュメントに表示する画像ファイルの名前を指定する。画像ファイルは、PICTフォーマットの他に、QuickTimeがインストールされているなら、

JPEGやGIFなどのQuickTimeが扱える画像フォーマットも利用できる。もちろん、Maxのサーチ・パス内に指定した画像ファイルが存在している必要がある。

pictオブジェクトには3つのインレットがあり、2番目のインレットで画像を表示する左端座標、3番目のインレットで上端座標を指定する。これらの座標指定を変えることで画像を移動させることができるが、幅や高さは指定できないため、結果として画像の縮小表示や拡大表示はできず、画像を保存した解像度のまま表示することになる。

画像を表示させるにはpictオブジェクトの第1インレットに1またはbangメッセージを送ればよい。また画像の表示位置を変えたい場合には、第1インレットに1またはbangメッセージを送ることで新しい位置に表示させることができる。画像を消去するには、第1インレットに0またはclearメッセージを送る。

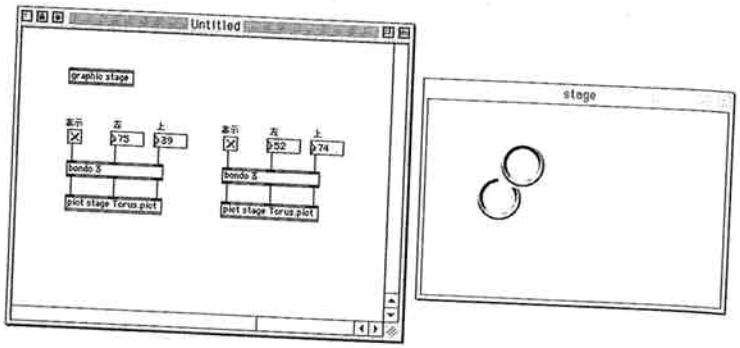
■6-4-7　pictオブジェクトによる画像の表示　ok 05/1/5



pictオブジェクトでは任意の画像を表示できるが、マスクや描画モードを指定することができない。そのため、複数の画像が重なった場合には、手前の画像の矩形領域が背後の画像を覆い隠してしまうことになる。

6-4-8の例では、トーラス(リング)上の画像だけでなく白い背景の部分も背後の画像を隠してしまっている。このような場合、画像が重ならないか、画像が重なっても不自然に見えない表現を工夫する必要がある。

■6-4-8 複数の画像が重なった場合の表示



05/1/5

└

略

picsオブジェクトによる画像の切り替え表示

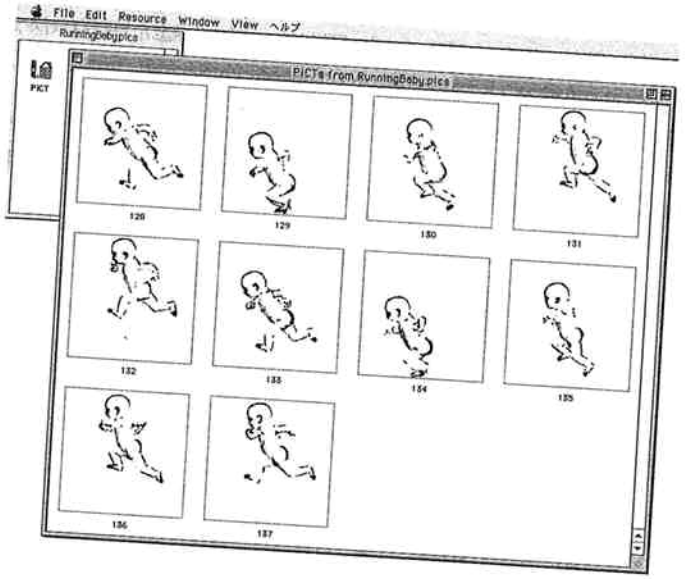
pictオブジェクトは特定の画像を表示するが、メッセージによって異なる画像を表示させることはできない。同じオブジェクトで複数の画像を切り替えて表示したい場合は、picsオブジェクトを用いなければならない。picsオブジェクトは、PICSファイルと呼ばれる特別なファイルを利用する。PICSファイルは複数のPICTリソースを持つので、1つのファイルの中に複数の画像を持たせることができる。フォーマットは異なるが、PICSファイルはAnimated GIFファイルと同じような画像ファイルと考えてよいだろう。

PICSファイルを作成できるアプリケーションは少ないが、APPLEが提供するリソース・エディターであるResEditを用いてPICSファイルを作成することができる。このためには、まず、ResEditで新しいファイルを作成する。そして、Photoshopなどのグラフィック・ソフトで作成した画像をクリップボードにコピーし、ResEditでペーストする。これで、ファイルに1つのPICTリソースが含まれることになる。

あとは必要な画像をコピーして、ResEditでペーストすることを順次繰り返せばよい。ここでは、子供が飛び跳ねるようにして走るアニメーションを10個のPICTリソースとして作成している。

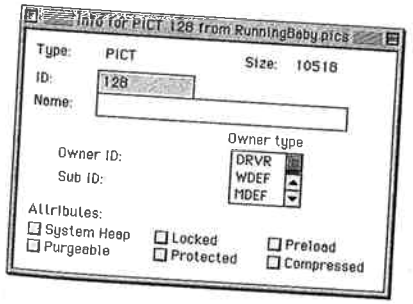
リソースにはID番号があり、ペーストを繰り返せば、128番から1つずつ大きな番号が付けられていく。通常はこれでよいが、PICTリソースを削除や追加する場合には、

■6-4-9 ResEditでのPICSファイルの作成



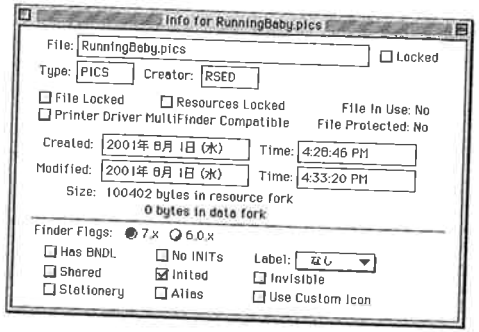
リソースID番号が連続するように注意しなければならない。リソースID番号を設定するには、リソースを選択し、ResEditのResourceメニューからGet Resource Infoを選ぶ。これでリソースの情報ウィンドウが開くので、そのID欄で設定できる。

■6-4-10 リソースの情報ウィンドウ



必要な数のPICTリソースを作成したら、ファイルを保存して閉じよう。次にファイルのタイプを指定しておかなければならない。このためには、ResEditのFileメニューからGet File/Folder Info...を選び、ファイル選択ダイアログで作成したPICSファイルを選ぶ。ファイルの情報ウィンドウが開けば、そのType欄をPICSとする。これで、ウィンドウを閉じて、ファイルの情報を保存する。このとき、日本語のOSを使用している場合は、日付のフォーマットが正しくないとのアラートが表示されるが、Yesボタンをクリックして、日付をそのままにして保存すればよい。

■6-4-11　ファイルの情報ウィンドウ

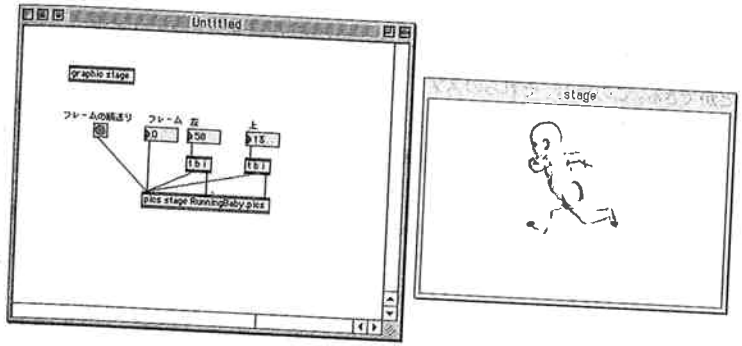


このようにしてPICSファイルを作成したら、さっそくpicsオブジェクトで表示させてみよう。picsオブジェクトの使用法はpictオブジェクトと似ている。picsオブジェクトの2番目のインレットで画像を表示する左端座標を指定し、3番目のインレットで上端座標を指定する。画像の表示位置の指定は、pictオブジェクトの第1インレットにbangメッセージを送った時点で反映される。

第1インレットに0を送れば、ID番号が最も小さいPICTリソースが画像として表示される。そして、1なら2番目のPICTリソース、2なら3番目のPICTリソース、という具合に第1インレットで受け取った整数に対応したPICTリソースの画像が表示される。また、第1インレットにbangメッセージを送れば、表示しているPICTリソースの次のPICTリソースが表示される。画像の表示を消去するには、第1インレットへclearメッセージを送ればよい。

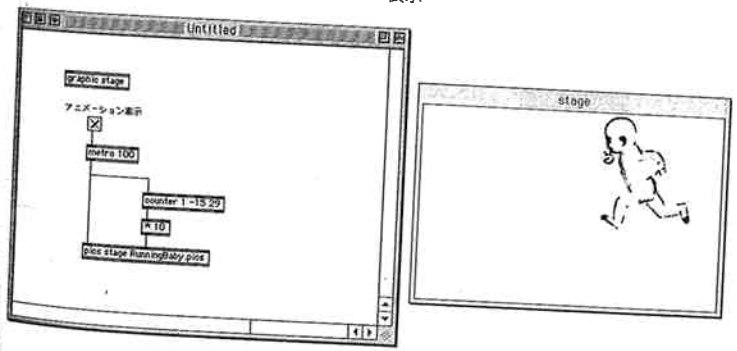
6-4-12のパッチでは、先ほど作ったPICSファイルをpicsオブジェクトで表示している。第1インレットにbangや整数メッセージを送れば、画像が切り替わるのが分かるだろう。また、左端座標や上端座標を設定するときに、同時に第1インレットにbangメッセージを送っているので、画像が変化しながら表示位置も変わることになる。

■6-4-12　picsオブジェクトによる画像表示



これをmetroオブジェクトを使って、一定時間間隔で自動的に画像が切り替わるようにしたのが、6-4-13のパッチだ。ここでは、左端座標も変化させて、子供が右から左へと走り抜けていくアニメーションにしている。

■6-4-13　picsオブジェクトを使ったアニメーション表示



なお、picsオブジェクトはpictオブジェクトと同じく、表示の幅や高さは指定できないため、画像の縮小表示や拡大表示はできない。また、マスクや描画モードを使って表示させることもできない。

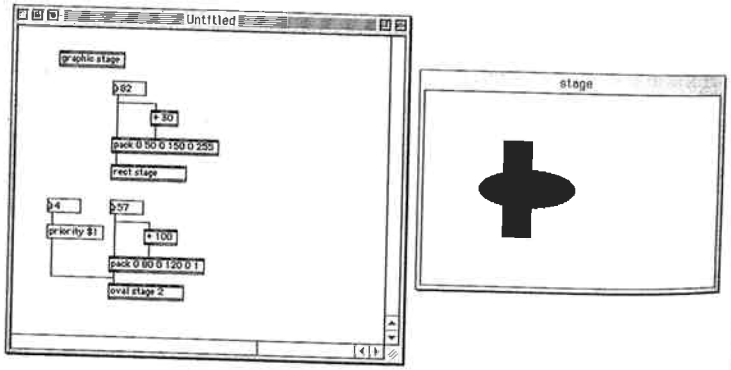
複数スプライトの処理

複数のオブジェクトを使えば、グラフィック・ウィンドウに複数のスプライトを表示することができる。この場合、それらのスプライトには前後関係が生まれる。

この前後関係はプライオリティと呼ばれ、オブジェクトの2番目のアーギュメントの整数として指定する。プライオリティの数値が大きいほど手前に表示される。プライオリティを指定しない場合は、デフォルト値として3が用いられる。また、priorityメッセージによって、プライオリティを変更することも可能だ。プライオリティが同じであれば、前後関係は特定できない。

6-4-14のパッチでは、rectオブジェクトはプライオリティを指定していないので、プライオリティは3として扱われる。これに対して、ovalオブジェクトは、アーギュメントでプライオリティを2に指定している。そのため、矩形と楕円形が重なる場合、楕円形の手前に矩形が表示されることになる。また、priorityメッセージによって楕円形のプライオリティを4以上にすれば、矩形より手前に楕円形が表示されるだろう。

■6-4-14 プライオリティによる図形の前後関係の設定



また、frame、rect、ring、ovalの各オブジェクトでは、第5インレットに受け取る数値によって、描画モードを設定できる。指定できる描画モードには次の種類がある。これはlcdの描画モードと同じだ。

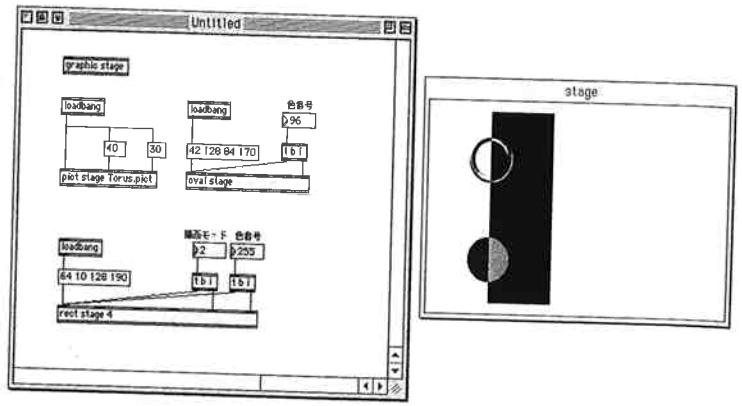
0	Copy
1	Or
2	Xor
3	Bic
4	NotCopy
5	NotOr
6	NotXor
7	NotBic
32	blend
33	addPin
34	addOver
35	subPin
36	transparent
37	adMax
38	subOver
39	adMin

描画モードの違いによって、スプライトが重なったときの表示が異なるので、6-4-15のようなパッチでどのような表示になるかを確認しておこう。

1つのオブジェクトが1つのスプライトを処理することは、メリットであり、デメリットでもある。例えば、いくつかのボールがウィンドウ内をはねるようなアニメーションをしたい場合は、それぞれのオブジェクトごとに処理をすれば、他のオブジェクトの状態を気にしなくても自動的に全体の表示が行われる。あるボールはゆっくり動き、他のボールは速く動いていても構わない。これは、それぞれが独立したオブジェクトであるためのメリットだろう。

一方、ボール同士が衝突してはね返るようにしたい場合は、処理が難しくなる。これ

■6-4-15 描画モードの違いによる描画



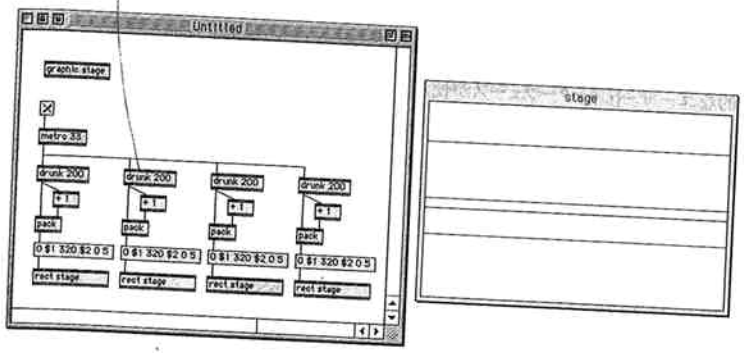
は、オブジェクト同士が連絡を取り合う方法がなく、すべてのオブジェクトを管理するオブジェクトもないからだ。そのため、graphic関連のオブジェクトは、それぞれが独立して動くアニメーションに適していると言えるだろう。

さらに、扱いたいスプライトの数が多い場合は、パッチの作成が面倒になる。6-4-16の例は、4本の横線がランダムに上下するパッチだ。ここでは、4本の横線を表現するために4つのrectオブジェクトを使っている。これが10本の線なら10個のrectオブジェクトが必要になり、関連するオブジェクトも同じだけ必要になる。パッチ・オブジェクトを用いると簡略化できるが、それでも10個のパッチ・オブジェクトを並べなければならない。さらに、数十本の線を描きたいとなると、現実的な作業ではなくなってくる。

このような場合に活用したいのがpoly~オブジェクトだ。poly~はMSPのオブジェクトで、本来ポリフォニックな発音のために設けられているが、これをMaxオブジェクトのために活用することもできる。

まず、1本の線がランダムに上下する処理をパッチ・オブジェクトとして作成しておこう。そして、poly~の最初のアーギュメントにパッチ・オブジェクトの名前を指定し、2番目のアーギュメントとして使用するパッチ・オブジェクトの数を指定する。ここでは50個のパッチ・オブジェクトを用いるように指定している。

■6-4-16 4本の線をランダムに動かす処理



これにより、50本の線が描かれ、それらは独立してランダムに上下に移動することになる。poly~オブジェクトを用いれば、簡潔なパッチで済むことが分かるだろう。poly~オブジェクトの詳しい使用方法については『Chapter 5 オーディオ処理』を参照してほしい。

■6-4-17 poly~オブジェクトによって50本の線をランダムに動かす処理