Bulletin of the Graduate School of Education and Human Development, Nagoya University (Psychology and Human Development Sciences) 2002, Vol. 49, 247 − 276.

心理学実験におけるMS-DOS（2）

── ビットマップ・ファイル表示ライブラリBMPLOADerの作成 ──[1)]

下木戸　隆司[2)]

前回（下木戸，2001）では，MS-DOSは過去の遺物などではなく，場合によってはむしろ優れた環境であることを論じた。MS-DOSはリアルタイム処理性に優れており，ミリ秒単位の精密な制御が容易であるためである。これらの利点はMicrosoft Windowsには継承されているとはいいがたいため，その点を考えるとMS-DOSの有用性は今日でも減じられていないといえる。しかしながら，MS-DOSで心理学実験を行うには今一つ利用しづらい側面があるのも否めないところがある。その一つにグラフィック・プログラムに関する敷居の高さが挙げられる。本稿ではそうしたグラフィック・プログラムを容易にする手段として，MS-DOS上で画像ファイルを表示させるやり方に着目する。その際に，PC-9800シリーズやPC/AT互換のコンピュータでグラフィックを表示させる場合の問題点，とくに心理学実験を行う場合に問題となる点についても議論を行う。

これまで心理学実験で図形やアニメーションを表示させる場合には，プログラム中で描画するというのが一般的であった。BASICやCといったプログラム言語は直線を引いたり，矩形部分を塗りつぶしたりする描画関数を備えているため，利用者はそれらを利用して意図通りのグラフィック・パターンを描くことができた。しかしこの方法は単純なグラフィック・パターンに対しては有効であるが，複雑なパターンを扱おうとする場合，描画に時間がかかり過ぎるという問題があった。グラフィック・パターンにフィルタリング処理を施して表示させたり，人物や風景などの写真画を表示させようとする場合には，多くの負荷がかかるため，あまり現実的なやり方ではなかった。またグラフィック・プログラムは実行速度を高めるために，ハードウェアに密着したプログラムを作成する（いわゆるハードウェアを“叩く”）必要があり，そのことが利用者の敷居を高いものにしていた。それらに加え，この方法では図形データがプログラム中に含まれることになるため，別の図形を用意する場合，その都度プログラムを書き換えなければならない問題もあった。

これらの問題を解決する一つの手段として，刺激を画像ファイル形式で予め作成しておき，実験ではそのファイルを読み込んで利用するというやり方がある。この方法なら複雑なグラフィック・パターンに対しても対応できるし，新たな画像ファイルを表示させる場合にもファイルを差し替えることで，プログラムを一々変更しなくても済む。画像ファイルを表示させる機能が提供されてしまえば，利用者はグラフィック表示装置固有の問題をとくに気に病む必要もなくなる。こういった利点のため，MEL（Schneider, 1988）やCedrus社のSuperLabをはじめとする多くの実験ソフトウェアでは，画像ファイルの表示が簡単にできるようになっている。本稿では，Windows環境で標準的な画像ファーマットであるビットマップ形式の画像ファイルに着目し，それをMS-DOS環境で表示させるライブラリ（BMPLOADer）について紹介する。BMPLOADerが提供する関数を利用することで，MS-DOS下での心理学実験の利点を活かしつつ，これまで障壁となっていたグラフィック・プログラミングの問題が改善されることが期待できる。BMPLOADerの機能に関して具体的に説明する前に，まずコンピュータでグラフィックがどのように扱われるのかに関して簡単に述べる。

1）BMPLOADerを使用することによって生じた不具合に対して，筆者は賠償及び保障に関する一切の責任を負い得ないのでご了承いただきたい。技術的な問い合わせやバグレポートは，電子メールにて筆者宛（E-mail: m47031a@cc.nagoya-u.ac.jp）に送付いただければ幸いである。なお，BMPLOADerの作成に関しては，BMP File Loader/Saver Ver. 1.1 for PC-9801 seriesのソース・ファイルが大いに参考になった。制作者のKan-chan氏に感謝いたします。

2）名古屋大学大学院教育発達科学研究科博士課程（後期課程）

グラフィック・パターンが画面に表示される仕組み

コンピュータはどのようにして画像を表示させているのだろうか。使用される装置は，映像を表示するグラフィック・ディスプレイ，画面に表示させる情報を保持しておくビデオ・メモリ（VRAM），描画環境の設定や図形の描画を行うグラフィック制御チップの三つに大別することができる（Figure 1 参照）[3]。これらの装置がうまく機能することによって，画面に文字や画像などのグラフィック・パターンが表示されるのである。これらの装置に関しては以下に説明する。

グラフィック・ディスプレイ　文字や画像などのグラフィック・パターンを画面に表示させる装置であり，単にディスプレイやモニタと呼ばれることが多い。伝統的にこれまではCRT（Cathode Ray Tube）ディスプレイが多く用いられてきたが，最近では液晶のものも普及してきている。

VRAM　VRAMは画面に表示される情報を保持するための装置であり，入出力を介してメモリの内容にアクセスできる。使用するコンピュータのグラフィック環境にもよるが，VRAMはたいていいくつかの区画に区分されており，それらの区画は互いに独立してアクセスすることが可能になっている。例えばPC-9800シリーズでは，VRAMはプレーンと呼ばれるメモリ区画を青，緑，赤，灰の四つ分備えている。PC-9800シリーズではこれらのプレーンをうまく使うことによってカラー表現を行っている（プレーンド・ピクセル方式）。VRAMの容量は，画面をどれだけ細かく表現できるかを示すドット解像度や，カラー表現数といったグラフィック性能に直結しているため，最近では大容量のものが用意されている。

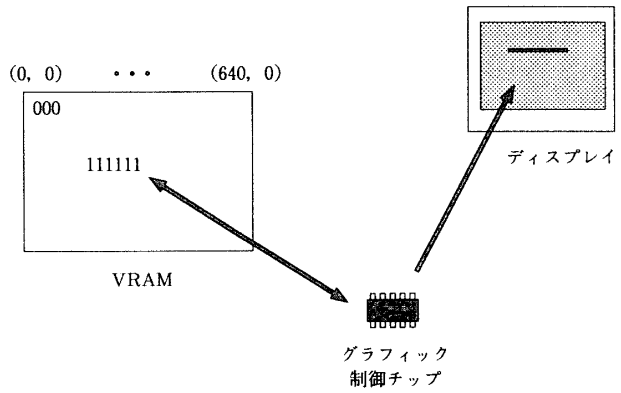

Figure 1　グラフィック表示装置に直線を表示させるまでの流れ

グラフィック制御チップ　グラフィック制御チップはGDC（Graphic Display Controller）やEGC（Enhanced Graphics Charger），あるいはEGA（Enhanced Graphics Adapter）やVGA（Video Graphics Array）に代表されるLSIの総称である。PC/AT互換機ではグラフィック環境の設定を行うグラフィックBIOSが本体内部の基盤に実装されないため，グラフィック制御チップはグラフィックBIOSを備えているのが普通である。グラフィック制御チップに期待されている役割は，基本的にはコンピュータの中央演算装置（Central Processing Unit: CPU）に代わって高速に描画を行うことである。Windowsのようなグラフィカルなインターフェースを有するOSでは，グラフィック演算は煩雑で膨大なものとなるため，それを受けて最近のチップは高性能なものが次々と登場している。

直線の描画

例えば，画面の任意の位置に直線を引きたいとする（Figure 1 参照）。意図通りに直線を描くためには，まず画面の座標を正確に把握する必要がある。画面の座標とVRAMのアドレスは対応しているため，画面のある位置に直線を描く場合にはその座標に対応するVRAMのアドレスに直線の情報を書き込まなければならない。縦横の二次元で表現される画面の座標とは異なり，VRAMのアドレスは一次元，つまり線形形式で表現されるので少しやっかいである。1行につき640ドットの解像度を持つPC-9800シリーズでは，X座標120，Y座標30に対応するのは，X＋80Y＝2,520，2,520番目のアドレスということになる。640Yではなく80Yとなっているのは，VRAMのアドレスが1バイト単位，つまり8ドットずつ区画されているためである。

書き込むアドレスがわかれば，次にそこに何を書き込むかである。書き込まれる情報は，非表示／表示を意味する0/1データであったり（プレーンド・ピクセル方式），書き込まれる色を示すパレット・レジスタの値であったり（パックド・ピクセル方式），使用するコンピュータのグラフィック環境によって異なっている。非表示の場合には背景と同じ色のドットが描かれていると見なせば，VRAMには色に関するデータが書き込まれていると考えることができる。例では，直線の線が描かれるアドレスには線の色に関するデータが，それ以外のアドレスには背景色に関するデータがVRAMに書き込まれること

3）ここではビデオ・カード（ビデオ・アダプタ），グラフィック・アクセラレータ，ビデオ・チップなどの装置を含む，広い意味でグラフィック制御チップを捉えている。

になる。これらのデータの書き込みはCPUが行うが，グラフィック制御チップが代わりに行うこともでき，その場合の方が描画速度が速くなる。

VRAMに書き込まれた情報が読み出され，ディスプレイに転送される。その際にVRAMに書き込まれていたデータは，RAMDAC（Random Access Memory Digital/Analog Converter）によってパレット・レジスタの内容に応じてアナログ形式に変換される。情報がアナログ形式に変換されることで，ディスプレイが表現できる色が飛躍的に増大するという利点がある。また送られた情報が実際に映像として反映されるタイミングには装置によって違いがあり，CRTディスプレイでは垂直同期信号に同調して表示されるという特徴がある。これらの段階を経て，直線がディスプレイに表示されることになる。普段，こうした一連の処理をとくに意識しなくてすんでいるのは，利用者のかわりにプログラムが自動的に行っているためである。

グラフィック環境におけるPC-9800シリーズとPC/AT互換機の相違点

基本的なグラフィック環境

PC-9800シリーズとPC/AT互換機はハードウェア構成で共通するところも多いが，それぞれ独自に発展してきた仕様と歴史を持っている。なかでも両者のグラフィック環境における違いは大きく，主にその理由のため，両者の間ではソフトウェア面での互換性がない。グラフィック環境について具体的に説明すると，まず標準的なPC-9800シリーズでは，16色の同時発色が可能である上，640ドット×400ドット分の解像度を備えている。その後，後継機種であるPC-9821シリーズでは640×480ドットの解像度，同時発色数は256色に強化されたが，基本的にPC-9800シリーズではグラフィック性能は固定化されており，機種による違いが小さいという特徴がある。それに対して，PC/AT互換機では機種によって，とくに内蔵されているグラフィック制御チップの性質によって，グラフィック性能が大きく変化するという特徴がある。いいかえればこれは，制御チップをより高性能なものと交換することで，PC/AT互換機のグラフィック性能が向上することを意味する。この柔軟性はPC/AT互換機の大きな特色であり，グラフィック機能を本体内部の基盤に実装させることで固定化させていたPC-9800シリーズとは大きく異なっている[4]。

PC/AT互換機では，VGAという事実上の標準規格となっているグラフィック環境がある。VGAはもともとIBM PS/2シリーズで実装されたグラフィック制御チップのことを指していたが，VGAの機能を継承した上で，それを拡張したチップがその後登場したことにより，今日では専ら規格の意味で用いられている。VGAをサポートするチップでは，640×480ドットの解像度，同時発色数256色での表示が保証される。最近のグラフィック制御チップは，VGAを超える高解像度（e.g., SVGA）でのフルカラー（16,777,216色）表示が可能になっており，それに伴ってMバイトを超える大容量のVRAMを備えているのがごく普通になっている。

このようにPC-9800シリーズとPC/AT互換機では，基本的なグラフィック環境が大きく異なるため，利用者は自分が使う機種の性質を把握しておくことが望ましい。グラフィック環境の違いが心理学実験にとってどう問題となるのかに関しては後ほど議論する。

日本語表示の仕組み

日本語処理の仕方はPC-9800シリーズとPC/AT互換機とでは大きく異なっているため，心理学実験で日本語を表示させる場合には注意が必要である。PC-9800シリーズは日本で開発されたこともあって，フォントなどのテキスト・データをROMデータとして基盤に内蔵している。そのため，高速な日本語表示を行うことが可能になっている。それに対し，もともとPC/AT互換機は外国で開発されたため，日本語表示の問題が必ずしも考慮されていないという問題がある。その一つとして，PC/AT互換機では日本語のテキスト・データはハード・ディスク内に格納されているという点が挙げられる。基盤とハード・ディスクではデータにアクセスする速さが大きく異なるために，PC/AT互換機で日本語を表示させるのは速度面でどうしても不利になってしまう。

またそれに加えて，PC/AT互換機では日本語が表示可能になっている状態，いわゆる日本語モードでは，プログラムがVRAMに直接アクセスする方式を採っていないという点にも留意する必要がある。日本語モード下ではVRAMは仮想化されており，VRAMへのアクセスは$DISP.SYSというドライバが行っている（服部，1995）。$DISP.SYSが提供する仮想VRAMにプログラムが情報を書き込み，それを$DISP.SYSが実際のVRAMに情報を書き込むことで画面に表示させている

4）余談ではあるが，PC-9800シリーズのこのような固定性はMS-DOSの時代にはさほど問題にならなかったものの，高性能のグラフィック性能を要求するWindowsが登場するや大きな障害となり，PC-9800シリーズが衰退する原因の一つにもなった。

わけである。この方式は，間にドライバが介在することで遅延が生じるという問題を有している。テキスト・データとして日本語を表示させる必要がないのであれば，場合によってはプログラムは英語モードで動作させた方がよいかもしれない。英語モードは直接VRAMにアクセスできるので，こうした遅延は生じないからである。PC/AT互換機での日本語モード/英語モードの切り替えはコマンドライン上で“chev jp”あるいは“chev us”を入力することで行う。

ハードウェアの性能が大きく向上した今日では，PC/AT互換機での日本語表示もだいぶ高速になったため，これらの点に関してはあまり問題にされなくなった。とはいえ，精密な心理学実験を行う上で，余計な遅延誤差を生み出す要因は少ないにこしたことはない。利用者はこれらのことを念頭に置いた上で実験を計画すべきであろう。

心理学実験での刺激提示の注意点

刺激提示を意図通りに行う際の問題としては，表示のタイミングと見かけの変化の二つが主に知られている（e.g., Dlhopolsky, 1989; Finley 1989; 舟川，1988; Graves & Bradley, 1987, 1988; Haussmann, 1992; Heathcote, 1988; Paredes, Miller, & Creeger, 1990; Segalowitz, 1987）。これらの要因は正確な実験実施を妨げるので，利用者はそれを念頭に置いて実験を行う必要がある。

表示のタイミングの問題

刺激表示のタイミングの問題は，表示装置としてCRTディスプレイを用いる場合と液晶ディスプレイを用いる場合で事情が異なってくる。これまで心理学実験ではCRTディスプレイが多く使用され，液晶ディスプレイはCRTディスプレイが適さない特殊な環境下での用途に限られていた。しかし近年では，液晶ディスプレイの技術革新と普及に伴い，表示装置として液晶ディスプレイが使用されるケースも増えてきている。

CRTディスプレイを表示装置として用いる際の注意点は舟川（1988）に詳しい。CRTは電子ビームを蛍光面に照射することでグラフィック・パターンを表示させている。電子ビームは常に照射されているわけではなく，定期的に発生する同期信号に合わせる形で，画面の左上部から右下部まで水平方向に照射される。正確にいえば，CRTディスプレイは十数ミリ秒に一回の割合で蛍光面を発光させることで映像を表現している。このような方式はインパルス型と呼ばれる。よくCRT画面がちらついて見えるのはこの特性のためである。心理学実験でCRTディスプレイを表示装置として使用する際には，垂直同期信号に同調させてグラフィック・パターンを表示させることが求められる。垂直同期信号発生のタイミングに合わない場合は，次の信号発生まで待たされることになり，その分の遅延を被ることになる。そのため提示時間に正確を期するのであれば，垂直同期信号の発生周期の倍数時間分しかCRTディスプレイは刺激を表示していない点に留意する必要がある。その際には発光の残像の影響を弱めるために，刺激のマスキングにも注意を払うべきである。そう考えれば，垂直同期信号の発生周期を下回る（例えば数ミリ秒程度）表示時間での刺激提示はCRTディスプレイでは不可能であることがわかる。

液晶ディスプレイは画面全体に配置された画素のオン／オフでドットを表現するため，CRTディスプレイのように描画のタイミングを同期信号に合わせなければならない必要性は生じない。しかし液晶ディスプレイにはそのかわりに，液晶の応答速度が遅いという問題がある。このため，動画などの大量のデータを短時間で処理しなければならない用途にはあまり適していない。十数ミリ秒程度の刺激提示を要する心理学実験に関しても同様である。液晶ディスプレイでは更新されるまで映像が画面に保持され続けており，このような方式はホールド型と呼ばれる。液晶ディスプレイで画面がにじんで見えることがあるのは主にこのためである。応答速度の遅さは液晶の特性上やむをえないところもあるが，最近の液晶ディスプレイには応答速度が大幅に改善されているものも出てきている。心理学実験の装置として液晶ディスプレイが採用されるケースは今後さらに増えていくだろう。その際には，利用者は自らが使用する液晶ディスプレイの応答速度がどれほどかを把握しておく必要がある。

見かけの変化の問題

文字や画像などのグラフィック・パターンは，それが表示されるグラフィック環境によって“見え”の様子が変わってくるという問題がある。これらの問題は光学的な表示装置固有のものであり，厳密に統制することは不可能といわざるをえない面もあるが，現実的に可能な面では注意するにこしたことはない。以下にこれらの問題点を列挙する。

大きさ　表示される画像のピクセル・サイズが同じでも，グラフィック環境が変わると，実際に表示される画像の大きさが変わってしまう。例えば，111×39ピクセルの長方形を17インチのCRTディスプレイに表示させる場合を考えてみる。このとき，実際に画面に表示される長方形の大きさは，ドット解像度やディスプレイの大きさ

によって異ってしまうのである。640×400ドットの環境では長方形はおよそ52mm×23mmの大きさになるが，320×200ドットの環境ではおよそ101mm×47mmの大きさになる。両者の間には約2倍の開きがある。また単に大きさが変わるだけでなく，それによって見かけの形が変わってしまう場合もある。元画像が正方形であったものが，別の画面では縦長の長方形になって見えたという話もよく聞くものであり，これもドット解像度やディスプレイの大きさでの相違が原因である。そのため，利用者は画像が実際にどのように表示されるのかを意識し，確認しておく必要がある。

輝度　グラフィック環境が変わるとディスプレイに表示されるグラフィック・パターンの輝度も違ってくる。輝度はディスプレイの仕様やその環境設定によるところが大きく，ブライトネスやガンマ補正等による調整の仕方によって変化する。多くの用途では目視による調整で問題ないと思われるが，厳密に輝度を統制する場合には測光計が必要である。

色度　表示される画像の色もグラフィック環境が変わると見かけが変わってしまう。色度もディスプレイのRGB輝度やガンマ補正の仕方に影響されるため，利用の際には意図通りの色で表現されているかを確認しておく必要がある。

ビットマップ・フォーマット

ここでは画像ファイルのビットマップ・フォーマットについて説明する。ビットマップ形式はWindows環境で標準的な画像フォーマットに位置づけられている。とくに256色のビットマップ・データはVGAをサポートするPC/AT互換機では扱いが容易なため，よく使用されている。ビットマップ形式自体は，2色，16色，256色，およびフルカラー（16,777,216色）に対応しており，基本的に圧縮されていない状態で利用されることが多いが，まれに圧縮されたものもある。未圧縮であるためにファイル・サイズが大きくなってしまうという欠点がある反面，データがシンプルなため，高速表示が可能という長所を持つ。圧縮形式の画像フォーマットとして利用されることの多いJPEG（Joint Photographic Coding Experts Group）形式などでは画像の復元に複雑な処理が要求されるため，使用するコンピュータの性能によっては表示が大分遅くなってしまうことがあるが，ビットマップ形式ではその問題はだいぶ軽減される。ただし，高速表示が可能というビットマップ形式の利点は現実面ではあまり大きなものではないかもしれない。というのは，心理学実験ではいきなり画像を読み込んで表示させるやり方よりも，いったんVRAMに格納しておいて，必要なときに画面を切り替えることで表示させるやり方を採ることの方が多いためである。その意味では，心理学実験でビットマップ形式が採用されることの利点は，単に扱い易さの点につきる。

次にビットマップのファイル構造について説明する。ビットマップ・ファイルはヘッダ部，カラー・インデックス部，イメージ部の三つの部分から構成される。ヘッダ部にはファイルがビットマップ形式であること示す標識（“BM”）や縦横のピクセル・サイズ，イメージ・サイズ（バイト単位），使用色等の情報が格納されている。カラー・インデックス部にはイメージで使用されている色のRGB値が8ビット単位で格納されている。いわゆる“パレット・データ”が格納されているといえばわかりやすいかもしれない。RGB値とは，光の三原色である赤（Red），緑（Green），青（Blue）の輝度を示すもので，この値を用いて無数の色調を表現することができる。これらのデータはビットマップ・ファイルが使用する色の数だけ含まれることになるが，例外的にフルカラーの場合は含まれない。この場合にはイメージ部にRGB値が直接記載されることになる。イメージ部は実際のイメージ・データが格納されている。原イメージの情報は左下から右上まで一列ずつ水平方向に格納される。実際にイメージ・データとして収められる情報は2色，16色，256色の場合はカラー・インデックス値であり，フルカラーの場合はRGB値である。このため1ピクセル当たりのデータ・サイズは，2色なら1ビット，16色なら4ビット，256色なら8ビット（1バイト），フルカラーなら24ビット（3バイト）というように，使用する色数に依存することになる。またビットマップには一列分の情報量は4バイトの倍数でなければならないという制約があり，これに充たない場合には画像データとしては特に意味を持たない詰め物（padding）がなされる。

BMPLOADer

BMPLOADerは，ビットマップ形式の画像ファイルをMS-DOS環境で表示させるための関数を格納したライブラリである。MS-DOSが動作する環境であれば，PC-9800シリーズとPC/AT互換機（ただしVGAに対応している必要がある）の双方で稼働する。BMPLOADerが提供する関数のうち主なものをTable 1に示しておく。

使用機種によって機能面で異なる箇所

PC-9800シリーズとPC/AT互換機とではグラフィック性能や仕様が大きく異なるため，BMPLOADerの機能を双方の環境で完全に等価にすることはできなかった。

Table 1　BMPLOADerが提供する主な関数

グラフィク環境設定	
void InitGScreen (void)	グラフィックの初期化処理（必須）
void CloseGScreen (void)	グラフィックの終了処理（必須）
void SetAPage (int page)	書き込む画面の指定
void SetVPage (int page)	表示する画面の指定
void ClearPage (int page)	画面の消去
void WaitVSync (void)	垂直同期信号発生待ち
カラー・パレットの設定	
void SetBMPPalette (unsigned char color, unsigned char red, unsigned char green, unsigned char blue, unsigned char bright)	
ビットマップ・ファイルの読み込み	
int LoadBMPFile (char *fname, int sx, int sy, unsigned char bright)	
画像データ取得	
int GetBMPXYSize (char *filename, int *xsize, int *ysize);	
int GetBMPSize (char *filename);	
int GetBMPColorBit (char *filename);	

Table 2　使用機種の違いによる機能の相違点

①ドット解像度	
PC-9800シリーズ	640×400
PC/AT互換機（VGA）	320×200
②表示可能なページ数	
PC-9800シリーズ	2枚
PC/AT互換機	4枚
③サポートするビットマップ画像の色数	
PC-9800シリーズ	2，16色
PC/AT互換機	2，16，256色

ここではこの相違点を簡単に述べる（Table 2 参照）。

まず，PC-9800シリーズとPC/AT互換機ではBMPLOADerが稼働するグラフィックの解像度が異なる。PC-9800シリーズ上では640×400ドットの環境で動作するが，PC/AT互換機上では320×200ドットの環境で動作する。この解像度の違いはグラフィック・パターンの見えの大きさの変化につながるため，注意が必要である。見かけの変化の問題に関しては先に論じたとおりである。

次に，PC-9800シリーズとPC/AT互換機ではBMPLOADerが使用できる画面のページ数が異なる。PC-9800シリーズでは2枚，PC/AT互換機（ただしVGAをサポートするものに限る）では4枚を表示することができる。表示可能なページ数が心理学実験にとって重要なのは，複数のページにそれぞれグラフィック・パターンをセットしておき，表示ページを瞬時に切り替えることで，ディスプレイをタキストスコープ的に利用することができるためである（e.g., Dlhopolsky, 1989; Finley, 1989）。利用者が使用できるページ数はVRAMの容量に依存するため，同じ容量であっても，より粗いドット解像度のもとで表示させたり，表示可能な色の数を減らしたりすることで，使えるページ数をさらに増やすことができる。実際にMyors（1998, 1999）は，表示可能なページをPC/AT互換機上で32〜240枚に増やす技法を紹介している。ところでPC-9800シリーズで表示可能なページ数が2枚しかないのはいかにも少なすぎるように思われるかもしれないが，この不足分はある程度なら，メモリ上に仮想画面を設けることによって補うことができる。画像情報をVRAMと同じ大きさを持つ配列のなかに予め書き込んでおいて，必要なときにそれをVRAMに転送させるというやり方である。これをハードウェア的に行う装置も製品化されている。

BMPLOADerが対応できるビットマップ画像の色数にもPC-9800シリーズとPC/AT互換機で違いがある。PC-9800シリーズでは2色，16色のビットマップを表示できるが，PC/AT互換機ではそれらに加えて，256色のビットマップも表示可能である。規定外の色数（例えばフルカラー）で保存されたビットマップにはエラーを返すので，このような画像を表示させたい場合は予め減

色処理を施しておく必要がある。

利用の仕方

BMPLOAD.Cファイル（付録1参照）をプログラム中でインクルードすることにより，BMPLOADerの関数を使用することができる。BMPLOADerはC言語コンパイラとしてBorland社のTurbo C/C++を想定して作成されているが，I/Oがらみの関数表現を修正すれば他のコンパイラでも利用可能かと思われる。またコンパイルする際には，マクロ・オプション（"－D"）を付けて，使用する機種がPC-9800シリーズであるのか，PC/AT互換機であるのかを明示する必要がある。これを忘れるとコンパイラからエラーを返されるので注意されたい。例えば，コンパイルするソース・ファイルが"SAMPLE.C"である場合を考えてみる。この場合，PC-9800シリーズであれば"TCC -DPC98 SAMPLE.C"（付録2参照）を，PC/AT互換機であれば"TCC -DDOSV SAMPLE.C"をコマンドライン上で入力することになる。とくに問題がなければ"SAMPLE.EXE"という実行ファイルが作成される。例ではプログラムをスモール・モデルで作成したが，BMPLOADerはすべてのメモリ・モデルに対応していることを付け加えておく。

使用上の注意

BMPLOADerはハードウェアの設定を人為的に変更するため，相性による問題が生じる可能性がある。とくにタイマ・ルーチンと併用する場合には注意されたい。タイマ動作中にファイル・アクセスすることは原則として望ましくないが，BMPLOADerを利用する際にはこの点が問題になりうる。画像データはテキスト・データと比べデータのサイズが大きい。そのため一度にデータをメモリ上に展開することができず，頻繁にファイル・アクセスを余儀なくされる場合があるためである。DOSTIMER（下木戸，2001）のようなソフトウェア・タイマは1ミリ秒ごとに割り込み処理を行っている。ファイル・アクセス中に割り込みがかかると，最悪の場合ファイルが破損する恐れがある。とくにファイル・アクセスに時間のかかるフロッピー・ディスクの場合は危険なので，ファイル・アクセスの際にはその都度タイマを止めておくことをおすすめする。

またPC-9800シリーズやPC/AT互換機の古い機種ではBMPLOADerが動作しない可能性がある。これらの機種ではBMPLOADerが動作の前提とするグラフィック環境を備えていないためである。IBM PC/XT シリーズのコンピュータ等がこれに該当する。

討　論

WindowsやMacintoshなどのグラフィカルなインターフェースを持つOSが普及することで，画像ファイルは以前に比べれば驚くほど身近なものとなった。MELやSuperLabに代表される実験ソフトウェアでも，画像ファイルの表示機能は標準的なものとして提供されている。それを受け，実験刺激として画像ファイルを作成しておき，MELやSuperLab等の実験ソフトウェアを利用して心理学実験を行うケースも増えてきている。しかし，これらの実験ソフトウェアは柔軟性に欠けるところがあり，意図通りの実験デザインが組めない場合も少なくないという問題がある。実験デザインの柔軟性を重視するのであれば，プログラム言語を使って実験を組むという方法も考えられるが，こうしたコンパイラには画像ファイルを表示させる関数を備えていないものが多く，プログラム中で画像ファイルを表示させるのが難しいという新たな問題が生じる。本稿で，画像ファイルの表示機能を提供するライブラリを紹介した理由は，画像ファイルを刺激として用いる実験に対しても，プログラム言語を使って実験デザインを組むことを支援するために他ならない。加えてそれがMS-DOS環境で動作するという利点もある。

MS-DOSはシンプルなOSであるために，意図通りの制御が容易という長所を持つ。実験の性質次第では，最適となる場合も少なくない。利用者は盲目的に一つの手段に固執するのではなく，実験の用途や自分のスキル等と照らし合わせて，実験手段として何を用いるのかをその都度検討することが望ましい。本稿で提案した方法がその際の選択肢の一つとなれば幸いである。

引　用　文　献

Dlhopolsky, J. G. 1989 Synchronizing stimulus displays with millisecond timer software for the IBM PC. *Behavior Research Methods, Instruments, & Computers,* 21, 441-446.

舟川政美　1988　実験制御技法とマシン語サブルーチン・プログラム集　PET　阿部純一（編）　パーソナル・コンピュータによる心理学実験プログラミング　ブレーン出版

Finley, G. P. 1989 Tachistoscopic software for the Hercules display controller. *Behavior Research Methods, Instruments, & Computers,* 21, 387-390.

Graves, R. & Bradley, R. 1987 Millisecond interval timer and auditory reaction time paradigms for the IBM PC. *Behavior Research Methods, Instruments, & Computers,* **19**, 30-35.

Graves, R. & Bradley, R. 1988 More on millisecond timing and tachistoscopic applications for the IBM PC. *Behavior Research Methods, Instruments, & Computers,* **20**, 408-412.

服部昌博 1995 DOS/VとC言語 ディスプレイ・ディスク入出力・ASYNC入出力編 工学図書株式会社

Haussmann, R. E. 1992 Tachistoscopic presentation and millisecond timing on the IBM PC/XT/AT and PS/2: A turbo Pascal unit to provide general-purpose routines for CGA, Hercules, EGA, and VGA monitors. *Behavior Research Methods, Instruments, & Computers,* **24**, 303-310.

Heathcote, A. 1988 Screen control and timing routines for the IBM microcomputer family using a high-level language. *Behavior Research Methods, Instruments, & Computers,* **20**, 289-297.

Myors, B. 1998 The PC tachistoscope has 32 pages. *Behavior Research Methods, Instruments, & Computers,* **30**, 457-461.

Myors, B. 1999 The PC tachistoscope has 240 pages. *Behavior Research Methods, Instruments, & Computers,* **31**, 329-333.

Paredes, D. R., Miller, K. E., & Creeger, C. 1990 Graphic precision: Controlling stimulus displays on IBM PC-compatible computer. *Behavior Research Methods, Instruments, & Computers,* **22**, 319-322.

Schneider, W. 1988 Micro Experimental Laboratory: An integrated system for IBM PC compatibles. *Behavior Research Methods, Instruments, & Computers,* **20**, 206-217.

Segalowitz, S. J. 1987 IBM PC tachistoscope: Text stimuli. *Behavior Research Methods, Instruments, & Computers,* **19**, 383-388.

下木戸隆司 1998 MS-DOS上で動くタイマ・ルーチン作成の試み 名古屋大学教育学研究科 教育心理学論集, **28**, 29-42.

下木戸隆司 2001 心理学実験におけるMS-DOS—ミリ秒単位のタイマ・ルーチンDosTimerを中心として—. 名古屋大学大学院教育発達科学研究科紀要（心理発達科学）, **48**, 315-341.

（2002年9月30日 受稿）

ABSTRACT

Using MS-DOS to conduct psychological experiments: II. Image files presentation.

Takashi SHIMOKIDO

This article suggested a method of displaying bitmap image files on MS-DOS. Using image files as experimental stimuli has some advantages. Because it reduced drawing processing load and enhanced usability benefit. Being called by C programs, this routines (i.e., BMPLOADer) provides users displaying function of bitmap images. Furthermore, problems of stimulus presentation in psychological experiments were also discussed.

付録 1　BMPLOAD.C（version 1.0）

```
/*============================================================================

        心理学実験用ビットマップ画像表示ルーチン
                BMPLOADer  version 1.0 for (Borland Turbo C)
                                2002/10/09   by Takashi Shimokido.

        コンパイル：
                PC-9800 シリーズの場合 tcc -DPC98 *.c
                PC/AT 互換機の場合              tcc -DDOSV *.c

============================================================================*/

#include      <stdio.h>
#include      <fcntl.h>
#include      <sys¥stat.h>
#include      <dos.h>
#include      <io.h>
#include      <dir.h>
#include      <string.h>
#include      <stdlib.h>
#include      <conio.h>

#define BUFSIZE 4096
#define PAGESIZE 16000
#define BMPID 0x4d42                                    /* 'BM' */


void bufinit(int fh);
int bufread(char *buf, unsigned int len);
void InitGScreen(void);
void setunchainedmode(void);
void CloseGScreen(void);
void SetAPage(int page);
void SetVPage(int page);
void ClearPage(int page);
void SetBMPPalette(unsigned char c, unsigned char r, unsigned char g,
                unsigned char b, unsigned char bright);
int dispbit2(int fh, int sx, int sy, unsigned char bright);
int dispbit4(int fh, int sx, int sy, unsigned char bright);
int dispbit8(int fh, int sx, int sy, unsigned char bright);
int LoadBMPFile(char *filename, int sx, int sy, unsigned char bright);
int GetBMPXYSize(char *filename, int *xsize, int *ysize);
int GetBMPSize(char *filename);
int GetBMPColorBit(char *filename);
void WaitVSync(void);


char FILEBUF[BUFSIZE];
int bufp=0, bufrestl=BUFSIZE;
int buffh;
int pagepos=0;

typedef struct {
        int bfType;
        long bfSize;
        int bfReserved1, bfReserved2;
        long bfOffBits;
} BITMAPFILEHEADER;
```

```
typedef struct {
        long biSize, biWidth, biHeight;
        int biPlanes, biBitCount;
        long biCompression;
        long biSizeImage, biXPelsPerMeter, biYPelsPerMeter;
        long biClrUsed, biClrImportant;
} BITMAPINFOHEADER;

typedef BITMAPFILEHEADER BF;
typedef BITMAPINFOHEADER BI;
BF bf;
BI bi;

typedef struct {
        unsigned char b, g, r, res;
} BMPPALETTE ;

BMPPALETTE pal[256];


/************************************************************

                        共通

************************************************************/

/*    ==== バッファの初期化  ====    */

void bufinit(int fh){

        bufp = 0;
        buffh = fh;
        bufrestl = BUFSIZE;

}


/*    ==== バッファの読み出し  ====    */

int bufread(char *buf, unsigned int len){

        int pos=0, l, restl, readl;

        if (bufp==0) if (read(buffh, FILEBUF, BUFSIZE)==-1) return(0);
        restl = len;
        bufrestl = BUFSIZE - bufp;

        if (bufrestl>=len){
                for (restl=len;restl>0;restl--){
                        *(buf+pos) = FILEBUF[bufp];
                        pos++;
                        bufp++;
                }
        }else{
                for (l=bufrestl;l>0;l--){
                        *(buf+pos) = FILEBUF[bufp];
                        pos++;
                        bufp++;
                }
                restl -= bufrestl;
```

```
                bufp = 0;
                do{
                        if ((bufrestl = read(buffh, FILEBUF, BUFSIZE))==-1) return(0);
                        if (restl>=BUFSIZE) readl = BUFSIZE;
                         else readl = restl;
                        for (l=readl;l>0;l--){
                                *(buf+pos) = FILEBUF[bufp];
                                pos++;
                                bufp++;
                        }
                        restl -= readl;
                } while (restl>0);
        }

        return(1);

}


/*************************************************************

                PC-9800 シリーズ用

*************************************************************/

#ifdef PC98

/*    ====  パレット変更  ====    */

void SetBMPPalette(unsigned char color, unsigned char r, unsigned char g,
                unsigned char b, unsigned char bright) {

        pal[color].r = r;
        pal[color].g = g;
        pal[color].b = b;
        outportb(0xa8, color);
        outportb(0xaa, g * bright / 1600);
        outportb(0xac, r * bright / 1600);
        outportb(0xae, b * bright / 1600);

}


/*    ====  グラフィックモード開始  ====    */

void InitGScreen(){

        union REGS inreg, outreg;

        inreg.h.ah = 0x42;
        inreg.h.ch = 0xc0;
        int86(0x18, &inreg, &outreg);              /* 400 ラインモード */

        SetVPage(0);                                       /* 0 ページ表示 */
        SetAPage(0);                                       /* 0 ページ書き込み */
        outportb(0x6a, 0x01);                      /* 16 色 */
        outportb(0xa2, 0x0d);                      /* グラフィック表示 */
        outportb(0x62, 0x0c);                      /* テキスト非表示 */

}
```

```
/*    ==== グラフィックモード終了 ====    */

void CloseGScreen(){

        outportb(0xa2, 0x0c);                    /* グラフィック非表示 */
        outportb(0x62, 0x0d);                    /* テキスト表示 */

}


/*    ==== 書き込むページの指定 ====    */

void SetAPage(int page){

        outportb(0xa6, (page%2));

}


/*    ==== 表示ページの指定 ====    */

void SetVPage(int page){

        outportb(0xa4, (page%2));
        while (!(inportb(0x60) & 0x20));    /* 垂直同期信号待ち */

}


/*    ==== 画面のクリア ====    */

void ClearPage(int page){

        long int far *bplane = (long int far *)0xa8000000L;
        long int far *rplane = (long int far *)0xb0000000L;
        long int far *gplane = (long int far *)0xb8000000L;
        long int far *iplane = (long int far *)0xe0000000L;
        int ofs;

        SetAPage(page);                                    /* ページ指定 */

        for (ofs=0;ofs<8000;ofs++){
                *bplane++ = 0;
                *rplane++ = 0;
                *gplane++ = 0;
                *iplane++ = 0;
        }
}


/*    ==== 2色データの表示 ====    */

int dispbit2(int fh, int sx, int sy, unsigned char bright){

        char far *bplane = (char far *)0xa8000000L;
        char far *rplane = (char far *)0xb0000000L;
        char far *gplane = (char far *)0xb8000000L;
        char far *iplane = (char far *)0xe0000000L;
        char far *bplanesave;
        char far *rplanesave;
```

```
char far *gplanesave;
char far *iplanesave;
int x, y, z, i, j, m;
char mask;
char data[4];
int xp_, yp;
int xs_, ys;
int xw_, yw;
int xe_, ye;
int xa_, ya;
int xlastdot;
unsigned char pale[4*2];

xs_ = ((sx + 4000) / 8) - 500;
xw_ = ((bi.biWidth - 1) / 32) + 1;
m = ((bi.biWidth - 1) % 32) / 8 + 1;
xlastdot  = ((bi.biWidth - 1) % 8) + 1;
xe_  = xs_ + xw_ - 1;
if ((xs_ > 79) || (xe_ < 0) || (xs_ > xe_)){
        close(fh);
        return(0);
}
xp_ = 0;
if (xs_ < 0){
        xp_ = -xs_;
        xs_ = 0;
}
xa_ = 0;
if (xe_ > 79){
        xa_ = xe_ - 79;
        xe_ = 79;
}
if (xa_ > 0) xlastdot = 8;
mask = 0xff << (8 - xlastdot);
xw_ = xe_ - xs_ + 1;

yw = bi.biHeight;
ys = sy + yw - 1;
ye  = sy;
if ((ys < 0) || (ye > 399) || (ys < ye)){
        close(fh);
        return(0);
}
yp = 0;
if (ys > 399){
        yp = ys - 399;
        ys = 399;
}
if (ye < 0) ye = 0;
yw = ys - ye + 1;

bufread(pale, 4*2);
for (i=0;i<2;i++){
        SetBMPPalette(i, pale[i*4+2], pale[i*4+1], pale[i*4], bright);
}

for (z=0;z<yp*(xp_+xw_+xa_);z++) bufread(data, 4);

bplane += ys*80 + xs_;
rplane += ys*80 + xs_;
gplane += ys*80 + xs_;
```

```
        iplane += ys*80 + xs_;
        for (y=0;y<yw;y++){

                for (z=0;z<xp_;z++) bufread(data, 4);

                bplanesave = bplane;
                rplanesave = rplane;
                gplanesave = gplane;
                iplanesave = iplane;
                for (x=0;x<xw_-1;x++){
                        bufread(data, 4);
                        for (j=0;j<4;j++){
                                *iplane++ = 0;
                                *bplane++ = data[j];
                                *gplane++ = 0;
                                *rplane++ = 0;
                        }
                }
                bufread(data, 4);
                for (j=0;j<m-1;j++){
                        *iplane++ = 0;
                        *bplane++ = data[j];
                        *gplane++ = 0;
                        *rplane++ = 0;
                }
                *iplane = (*iplane & ~mask);
                *bplane = (data[m-1] & mask) | (*bplane & ~mask);
                *gplane = (*gplane & ~mask);
                *rplane = (*rplane & ~mask);

                for (z=0;z<xa_;z++) bufread(data, 4);

                bplane = bplanesave - 80;
                rplane = rplanesave - 80;
                gplane = gplanesave - 80;
                iplane = iplanesave - 80;
        }

        return(1);

}


/*    ====  16色データの表示  ====    */

int dispbit4(int fh, int sx, int sy, unsigned char bright){

        char far *bplane = (char far *)0xa8000000L;
        char far *rplane = (char far *)0xb0000000L;
        char far *gplane = (char far *)0xb8000000L;
        char far *iplane = (char far *)0xe0000000L;
        char far *bplanesave;
        char far *rplanesave;
        char far *gplanesave;
        char far *iplanesave;
        int x, y, z, i;
        char mask;
        char d1, d2, d3, d4;
        char data[4];
        int xp_, yp;
        int xs_, ys;
```

```
int xw_, yw;
int xe_, ye;
int xa_, ya;
int xlastdot;
unsigned char pale[4*16];

xs_ = ((sx + 4000) / 8) - 500;
xw_ = ((bi.biWidth - 1) / 8) + 1;
xlastdot  = ((bi.biWidth - 1) % 8) + 1;
xe_   = xs_ + xw_ - 1;
if ((xs_ > 79) || (xe_ < 0) || (xs_ > xe_)){
        close(fh);
        return(0);
}
xp_ = 0;
if (xs_ < 0){
        xp_ = -xs_;
        xs_ = 0;
}
xa_ = 0;
if (xe_ > 79){
        xa_ = xe_ - 79;
        xe_ = 79;
}
if (xa_ > 0) xlastdot = 8;
mask = 0xff << (8-xlastdot);
xw_ = xe_ - xs_ + 1;

yw = bi.biHeight;
ys = sy + yw - 1;
ye   = sy;
if ((ys < 0) || (ye > 399) || (ys < ye)){
        close(fh);
        return(0);
}
yp = 0;
if (ys > 399){
        yp = ys - 399;
        ys = 399;
}
if (ye < 0) ye = 0;
yw = ys - ye + 1;

bufread(pale, 4*16);
for (i=0;i<16;i++){
        SetBMPPalette(i, pale[i*4+2], pale[i*4+1], pale[i*4], bright);
}

for (z=0;z<yp*(xp_+xw_+xa_);z++) bufread(data, 4);

bplane += ys*80 + xs_;
rplane += ys*80 + xs_;
gplane += ys*80 + xs_;
iplane += ys*80 + xs_;
for (y=0;y<yw;y++){

        for (z=0;z<xp_;z++) bufread(data, 4);

        bplanesave = bplane;
        rplanesave = rplane;
        gplanesave = gplane;
```

```
            iplanesave = iplane;
            for (x=0;x<xw_-1;x++){
                  bufread(data, 4);
                  d1 = (data[0] & 0x80) | ((data[0] & 0x08) << 3) |
                        ((data[1] & 0x80) >> 2) | ((data[1] & 0x08) << 1) |
                        ((data[2] & 0x80) >> 4) | ((data[2] & 0x08) >> 1) |
                        ((data[3] & 0x80) >> 6) | ((data[3] & 0x08) >> 3);
                  d2 =((data[0] & 0x40) << 1) | ((data[0] & 0x04) << 4) |
                        ((data[1] & 0x40) >> 1) | ((data[1] & 0x04) << 2) |
                        ((data[2] & 0x40) >> 3) | (data[2] & 0x04) |
                        ((data[3] & 0x40) >> 5) | ((data[3] & 0x04) >> 2);
                  d3 =((data[0] & 0x20) << 2) | ((data[0] & 0x02) << 5) |
                        (data[1] & 0x20) | ((data[1] & 0x02) << 3) |
                        ((data[2] & 0x20) >> 2) | ((data[2] & 0x02) << 1) |
                        ((data[3] & 0x20) >> 4) | ((data[3] & 0x02) >> 1);
                  d4 =((data[0] & 0x10) << 3) | ((data[0] & 0x01) << 6) |
                        ((data[1] & 0x10) << 1) | ((data[1] & 0x01) << 4) |
                        ((data[2] & 0x10) >> 1) | ((data[2] & 0x01) << 2) |
                        ((data[3] & 0x10) >> 3) | (data[3] & 0x01);
                  *iplane++ = d1;
                  *gplane++ = d2;
                  *rplane++ = d3;
                  *bplane++ = d4;
            }
            bufread(data, 4);
            d1 = (data[0] & 0x80) | ((data[0] & 0x08) << 3) |
                  ((data[1] & 0x80) >> 2) | ((data[1] & 0x08) << 1) |
                  ((data[2] & 0x80) >> 4) | ((data[2] & 0x08) >> 1) |
                  ((data[3] & 0x80) >> 6) | ((data[3] & 0x08) >> 3);
            d2 =((data[0] & 0x40) << 1) | ((data[0] & 0x04) << 4) |
                  ((data[1] & 0x40) >> 1) | ((data[1] & 0x04) << 2) |
                  ((data[2] & 0x40) >> 3) | (data[2] & 0x04) |
                  ((data[3] & 0x40) >> 5) | ((data[3] & 0x04) >> 2);
            d3 =((data[0] & 0x20) << 2) | ((data[0] & 0x02) << 5) |
                  (data[1] & 0x20) | ((data[1] & 0x02) << 3) |
                  ((data[2] & 0x20) >> 2) | ((data[2] & 0x02) << 1) |
                  ((data[3] & 0x20) >> 4) | ((data[3] & 0x02) >> 1);
            d4 =((data[0] & 0x10) << 3) | ((data[0] & 0x01) << 6) |
                  ((data[1] & 0x10) << 1) | ((data[1] & 0x01) << 4) |
                  ((data[2] & 0x10) >> 1) | ((data[2] & 0x01) << 2) |
                  ((data[3] & 0x10) >> 3) | (data[3] & 0x01);
            *iplane = (d1 & mask) | (*iplane & ~mask);
            *gplane = (d2 & mask) | (*gplane & ~mask);
            *rplane = (d3 & mask) | (*rplane & ~mask);
            *bplane = (d4 & mask) | (*bplane & ~mask);

            for (z=0;z<xa_;z++) bufread(data, 4);

            bplane = bplanesave - 80;
            rplane = rplanesave - 80;
            gplane = gplanesave - 80;
            iplane = iplanesave - 80;
      }

      return(1);

}

/*    ==== ビットマップ・ファイルの読み込み  ====    */
```

```
int LoadBMPFile(char *filename, int sx, int sy, unsigned char bright){

        char hbuf[sizeof(BF)+sizeof(BI)];
        int fh;

        if ((fh = open(filename,O_RDONLY|O_BINARY)) == -1){
                close(fh);
                return(0);
        }

        bufinit(fh);
        if (!bufread(hbuf, sizeof(BF)+sizeof(BI))){
                close(fh);
                return(0);
        }

        bf = *((BF *)hbuf);
        bi = *((BI *)(hbuf+sizeof(BF)));
        if (bf.bfType != (int)BMPID){
                close(fh);
                return(0);
        }

        switch (bi.biBitCount){
                case 1:
                        dispbit2(fh, sx, sy, bright);
                        break;
                case 4:
                        dispbit4(fh, sx, sy, bright);
                        break;

                default:
                        close(fh);
                        return(0);
        }

        close(fh);
        return(1);

}


/*    ====  垂直同期信号待ち  ====    */

void WaitVSync(void){

        while ((inportb(0x60) & 0x20));
        while (!(inportb(0x60) & 0x20));

}

#endif


/*************************************************************

                        PC/AT 互換機用

*************************************************************/

#ifdef DOSV
```

```
void SetBMPPalette(unsigned char color, unsigned char r, unsigned char g,
                   unsigned char b, unsigned char bright) {

        pal[color].r = r;
        pal[color].g = g;
        pal[color].b = b;
        outportb(0x03c8, color);
        outportb(0x03c9, r*bright/400);
        outportb(0x03c9, g*bright/400);
        outportb(0x03c9, b*bright/400);

}


/*    ====  グラフィックモード開始  ====    */

void InitGScreen(void){

        union REGS inreg, outreg;

        inreg.h.ah = 0x00;
        inreg.h.al = 0x13;
        int86(0x10, &inreg, &outreg);              /* 256 色グラフィックモード */

        setunchainedmode();                                /* unchained モード(mode x) */

}


/*    ====  unchained mode 開始  ====    */

void setunchainedmode(void){

        int i;
        long int far *plane = (long int far *)0xa0000000L;

        outport(0x03c4, 0x04);                             /* chain-4 モード解除 */
        outport(0x03c5, 0x06);

        outport(0x03c4, 0xff02);                  /* すべてのプレーンにマスク */

        for(i=0;i<0x4000;i++){
                *plane++ = 0;                              /* メモリー初期化 */
        }

        outportb(0x03d4, 0x14);                            /* ロングモード解除 */
        outportb(0x03d5, 0x00);

        outportb(0x03d4, 0x17);                            /* バイトモード設定 */
        outportb(0x03d5, 0xe3);

}


/*    ====  グラフィックモード終了  ====    */

void CloseGScreen(void){

        union REGS inreg, outreg;

        inreg.h.ah = 0x00;
```

```
        inreg.h.al = 0x03;
        int86(0x10, &inreg, &outreg);              /* テキストモード */

}


/*    ==== 書き込むページの指定 ====    */

void SetAPage(int page){

        pagepos = (page%4) * PAGESIZE;

}


/*    ==== 表示ページの指定 ====    */

void SetVPage(int page){

        unsigned char far *plane = (unsigned char far *)0xa0000000L;
        unsigned int start_address, high_address, low_address;

        start_address = (int)plane + (page%4) * PAGESIZE;
        high_address = (start_address & 0xff00) | 0x0c;
        low_address = (start_address << 8) | 0x0d;

        while ((inp(0x03da) & 0x01));
        while (!(inp(0x03da) & 0x08));             /* 垂直同期信号待ち */

        disable();
        outport(0x03d4, high_address);             /* 表示アドレス指定 */
        outport(0x03d4, low_address);
        enable();

}


/*    ==== 画面のクリア ====    */

void ClearPage(int page){

        long int far *plane = (long int far *)0xa0000000L;
        int start_address, ofs;

        start_address = (page%4) * 4000;
        plane += start_address;

        outport(0x03c4, 0xff02);                   /* すべてのプレーンにマスク */
        for (ofs=0;ofs<4000;ofs++){
                *plane++ = 0;                                  /* メモリー初期化 */
        }

}


/*    ==== 2色データの表示 ====    */

int dispbit2(int fh, int sx, int sy, unsigned char bright){

        unsigned char far *plane = (unsigned char far *)0xa0000000L;
        unsigned char far *planesave;
```

```
int x, y, z, i, j, k, m;
unsigned char data[4];
unsigned char d[16], dd;
char mask;
int xp_, yp;
int xs_, ys;
int xw_, yw;
int xe_, ye;
int xa_, ya;
int xlastdot;
unsigned char pale[4*2];

xs_ = sx / 4;
xw_ = ((bi.biWidth - 1) / 32) + 1;
m = ((bi.biWidth - 1) % 32) / 8 + 1;
xlastdot  = ((bi.biWidth - 1) % 8) + 1;
xe_ = xs_ + xw_ - 1;
if ((xs_ > 79) || (xe_ < 0) || (xs_ > xe_)){
        close(fh);
        return(0);
}
xp_ = 0;
if (xs_ < 0){
        xp_ = -xs_;
        xs_ = 0;
}
xa_ = 0;
if (xe_ > 79){
        xa_ = xe_ - 79;
xe_ = 79;
}
if (xa_ > 0) xlastdot = 0;
mask = 0xff << (8-xlastdot);
xw_ = xe_ - xs_ + 1;

yw = bi.biHeight;
ys = sy + yw - 1;
ye   = sy;
if ((ys < 0) || (ye > 199) || (ys < ye)){
        close(fh);
        return(0);
}
yp = 0;
if (ys > 199){
        yp = ys - 199;
        ys = 199;
}
if (ye < 0) ye = 0;
yw = ys - ye + 1;

bufread(pale, 4*2);
outportb(0x03c6, 0xff);                              /* prepare color palette */
for (i=0;i<2;i++){
        SetBMPPalette(i, pale[i*4+2], pale[i*4+1], pale[i*4], bright);
}

for (z=0;z<yp*(xp_+xw_+xa_);z++) bufread(data, 4);

plane += ys*80 + xs_ + pagepos;
for (y=0;y<yw;y++){
```

```
                for (z=0;z<xp_;z++) bufread(data, 4);

                planesave = plane;
                for (x=0;x<xw_-1;x++){
                        bufread(data, 4);
                        for (j=0;j<4;j++){
                                for (k=7;k>=0;k--){
                                        d[j*8+(7-k)] = (data[j] & (1 << k)) >> k;
                                }
                        }
                        for (j=0;j<4;j++){
                                outport(0x03c4, ((1 << (j%4)) << 8) | 0x02);
                                for (k=0;k<8;k++){
                                        *plane++ = d[j+4*k];
                                }
                                plane -= 8;
                        }
                        plane += 8;

                }
                bufread(data, 4);
                for (j=0;j<4;j++){
                        for (k=7;k>=0;k--){
                                d[j*8+(7-k)] = (data[j] & (1 << k)) >> k;
                        }
                }
                dd = ((data[m-1] & mask) | (*plane & ~mask));
                for (k=7;k>=0;k--){
                        d[(m-1)*8+(7-k)] = (dd & (1 << k)) >> k;
                }
                for (j=0;j<4;j++){
                        outport(0x03c4, ((1 << (j%4)) << 8) | 0x02);
                        for (k=0;k<8;k++){
                                *plane++ = d[j+4*k];
                        }
                        plane -= 8;
                }

                for (z=0;z<xa_;z++) bufread(data, 4);

                plane = planesave - 80;
        }

        return(1);

}


/*    ==== 16色データの表示 ====    */

int dispbit4(int fh, int sx, int sy, unsigned char bright){

        unsigned char far *plane = (unsigned char far *)0xa0000000L;
        unsigned char far *planesave;
        int x, y, z, i, j, k;
        unsigned char data[4];
        unsigned char d[8], dd;
        char mask;
        int xp_, yp;
        int xs_, ys;
        int xw_, yw;
```

```
int xe_, ye;
int xa_, ya;
int xlastdot;
unsigned char pale[4*16];

xs_ = sx / 4;
xw_ = ((bi.biWidth - 1) / 8) + 1;
k = ((bi.biWidth - 1) % 8) + 1;
xlastdot = k;
xe_ = xs_ + xw_ - 1;
if ((xs_ > 79) || (xe_ < 0) || (xs_ > xe_)){
        close(fh);
        return(0);
}
xp_ = 0;
if (xs_ < 0){
        xp_ = -xs_;
        xs_ = 0;
}
xa_ = 0;
if (xe_ > 79){
        xa_ = xe_ - 79;
        xe_ = 79;
}
if (xa_ > 0) xlastdot = 0;
mask = (xlastdot % 2) << 4;
mask = 0xff << mask;
xw_ = xe_ - xs_ + 1;

yw = bi.biHeight;
ys = sy + yw - 1;
ye   = sy;
if ((ys < 0) || (ye > 199) || (ys < ye)){
        close(fh);
        return(0);
}
yp = 0;
if (ys > 199){
        yp = ys - 199;
        ys = 199;
}
if (ye < 0) ye = 0;
yw = ys - ye + 1;

bufread(pale, 4*16);
outportb(0x03c6, 0xff);                                  /* prepare color palette */
for (i=0;i<16;i++){
        SetBMPPalette(i, pale[i*4+2], pale[i*4+1], pale[i*4], bright);
}

for (z=0;z<yp*(xp_+xw_+xa_);z++) bufread(data, 4);

plane += ys*80 + xs_ + pagepos;
for (y=0;y<yw;y++){

        for (z=0;z<xp_;z++) bufread(data, 4);

        planesave = plane;
        for (x=0;x<xw_-1;x++){
                bufread(data, 4);
                for (j=0;j<8;j+=2){
```

```
                        d[j] = data[j>>1] >> 4;
                        d[j+1] = data[j>>1] & 0xf;
                }
                for (j=0;j<4;j++){
                        outport(0x03c4, ((1 << (j%4)) << 8) | 0x02);
                        *plane++ = d[j];
                        *plane-- = d[j+4];
                }
                plane += 2;
        }
        bufread(data, 4);
        for (j=0;j<8;j+=2){
                d[j] = data[j>>1] >> 4;
                d[j+1] = data[j>>1] & 0xf;
        }
        dd = data[(k-1)/2];
        d[k-1] = ((dd & mask) | (*plane & ~mask)) >> 4;
        d[k] = ((dd & mask) | (*plane & ~mask)) & 0xf;
        for (j=0;j<4;j++){
                outport(0x03c4, ((1 << (j%4)) << 8) | 0x02);
                *plane++ = d[j];
                *plane-- = d[j+4];
        }

        for (z=0;z<xa_;z++) bufread(data, 4);

        plane = planesave - 80;
    }

    return(1);

}


/*   ==== 256色データの表示 ====   */

int dispbit8(int fh, int sx, int sy, unsigned char bright){

    unsigned char far *plane = (unsigned char far *)0xa0000000L;
    unsigned char far *planesave;
    int x, y, z, i, j, k;
    unsigned char data[4];
    int xp_, yp;
    int xs_, ys;
    int xw_, yw;
    int xe_, ye;
    int xa_, ya;
    unsigned char pale[4*256];

    xs_ = sx / 4;
    xw_ = ((bi.biWidth - 1) / 4) + 1;
    k = ((bi.biWidth - 1) % 4) + 1;
    xe_  = xs_ + xw_ - 1;
    if ((xs_ > 79) || (xe_ < 0) || (xs_ > xe_)){
        close(fh);
        return(0);
    }
    xp_ = 0;
    if (xs_ < 0){
        xp_ = -xs_;
        xs_ = 0;
```

```
        }
        xa_ = 0;
        if (xe_ > 79){
                xa_ = xe_ - 79;
                xe_ = 79;
        }
        xw_ = xe_ - xs_ + 1;

        yw = bi.biHeight;
        ys = sy + yw - 1;
        ye  = sy;
        if ((ys < 0) || (ye > 199) || (ys < ye)){
                close(fh);
                return(0);
        }
        yp = 0;
        if (ys > 199){
                yp = ys - 199;
                ys = 199;
        }
        if (ye < 0) ye = 0;
        yw = ys - ye + 1;

        bufread(pale, 4*256);
        outportb(0x03c6, 0xff);                          /* prepare color palette */
        for (i=0;i<256;i++){
                SetBMPPalette(i, pale[i*4+2], pale[i*4+1], pale[i*4], bright);
        }

        for (z=0;z<yp*(xp_+xw_+xa_);z++) bufread(data, 4);

        plane += ys*80 + xs_ + pagepos;
        for (y=0;y<yw;y++){

                for (z=0;z<xp_;z++) bufread(data, 4);

                planesave = plane;

                for (x=0;x<xw_-1;x++){
                        bufread(data, 4);
                        for (j=0;j<4;j++){
                                outport(0x03c4, ((1 << (j%4)) << 8) | 0x02);
                                *plane = data[j];
                        }
                        plane++;
                }
                bufread(data, 4);
                for (j=0;j<k;j++){
                        outport(0x03c4, ((1 << (j%4)) << 8) | 0x02);
                        *plane = data[j];
                }

                for (z=0;z<xa_;z++) bufread(data, 4);

                plane = planesave - 80;
        }

        return(1);
}
```

```
/*    ====  ビットマップ・ファイルの読み込み  ====    */

int LoadBMPFile(char *filename, int sx, int sy, unsigned char bright){

        char hbuf[sizeof(BF)+sizeof(BI)];
        int fh;

        if ((fh = open(filename,O_RDONLY|O_BINARY)) == -1){
                close(fh);
                return(0);
        }

        bufinit(fh);
        if (!bufread(hbuf, sizeof(BF)+sizeof(BI))){
                close(fh);
                return(0);
        }

        bf = *((BF *)hbuf);
        bi = *((BI *)(hbuf+sizeof(BF)));
        if (bf.bfType != (int)BMPID){
                close(fh);
                return(0);
        }

        switch (bi.biBitCount){
                case 1:
                        dispbit2(fh, sx, sy, bright);
                        break;
                case 4:
                        dispbit4(fh, sx, sy, bright);
                        break;
                case 8:
                        dispbit8(fh, sx, sy, bright);
                        break;

                default:
                        close(fh);
                        return(0);
        }

        close(fh);
        return(1);

}


/*    ====  垂直同期信号待ち  ====    */

void WaitVSync(void){

        while ((inportb(0x03da) & 0x08));
        while (!(inportb(0x03da) & 0x08));

}


#endif
```

```
/************************************************************

                        共通

************************************************************/

/*    ==== 画像縦横サイズ取得  ====    */

int GetBMPXYSize(char *filename, int *xsize, int *ysize){

        char hbuf[sizeof(BF)+sizeof(BI)], i;
        int fh;

        if ((fh = open(filename,O_RDONLY|O_BINARY)) == -1){
                close(fh);
                return(0);
        }

        if (read(fh, hbuf, sizeof(BF)+sizeof(BI)) == -1){
                close(fh);
                return(0);
        }

        bf = *((BF *)hbuf);
        bi = *((BI *)(hbuf+sizeof(BF)));
        if (bf.bfType != (int)BMPID){
                close(fh);
                return(0);
        }

        *xsize = (int)(bi.biWidth);
        *ysize = (int)(bi.biHeight);

        return(1);

}


/*    ==== 画像サイズ取得  ====    */

int GetBMPSize(char *filename){

        char hbuf[sizeof(BF)+sizeof(BI)], i;
        int imagesize;
        int fh;

        if ((fh = open(filename,O_RDONLY|O_BINARY)) == -1){
                close(fh);
                return(0);
        }

        if (read(fh, hbuf, sizeof(BF)+sizeof(BI)) == -1){
                close(fh);
                return(0);
        }

        bf = *((BF *)hbuf);
        bi = *((BI *)(hbuf+sizeof(BF)));
        if (bf.bfType != (int)BMPID){
                close(fh);
                return(0);
```

```
        }

        imagesize = (int)(bi.biSize);
        return(imagesize);

}


/*    ==== 画像使用色数取得  ====    */

int GetBMPColorBit(char *filename){

        char hbuf[sizeof(BF)+sizeof(BI)], i;
        int colors;
        int fh;

        if ((fh = open(filename,O_RDONLY|O_BINARY)) == -1){
                close(fh);
                return(0);
        }

        if (read(fh, hbuf, sizeof(BF)+sizeof(BI)) == -1){
                close(fh);
                return(0);
        }
```

付録 2　SAMPLE.C

```
/*-------------------------------------------------------------------------

        BMPLOADer のサンプル
                                2002/09/28   by Takashi Shimokido.

        矢印キー（↑ ↓）で２枚の画像（ONIGIRI.BMP, NEKO.BMP）を切り替えます。
        エスケープ・キーで終了します。

        コンパイル：
                PC-9800 シリーズの場合  TCC -DPC98 SAMPLE.C
                PC/AT 互換機の場合              TCC -DDOSV SAMPLE.C

-------------------------------------------------------------------------*/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#include "bmpload.c"


#ifdef PC98

 #define XSIZE 640
 #define YSIZE 400
 #define UP 0x0b
 #define DOWN 0x0a
 #define ESCAPE 27
 #define PNUM 16
```

```
#endif

#ifdef DOSV

 #define XSIZE 320
 #define YSIZE 200
 #define UP 72
 #define DOWN 80
 #define ESCAPE 27
 #define PNUM 256

#endif


int main(void);


int main(void) {

    int i, c, x, y, xs, ys, tm, page = 0;
    char fname1[16]="onigiri.bmp";
    char fname2[16]="neko.bmp";
    BMPPALETTE pagepal1[256], pagepal2[256];

        InitGScreen();

/* 第1ページへの書き込み */
    SetAPage(1);
    GetBMPXYSize(fname2, &xs, &ys);
    x=(XSIZE-xs)/2;
    y=(YSIZE-ys)/2;
    tm = LoadBMPFile(fname2, x, y, 100);
    if (tm!=1){
        CloseGScreen();
        fprintf(stderr, "画像の読み込みに失敗しました。¥n");
        return(-1);
    }
    for (i=0;i<PNUM;i++){
        pagepal2[i].r = pal[i].r;
        pagepal2[i].g = pal[i].g;
        pagepal2[i].b = pal[i].b;
    }

/* 第0ページへの書き込み */
    SetAPage(0);
    GetBMPXYSize(fname1, &xs, &ys);
    x=(XSIZE-xs)/2;
    y=(YSIZE-ys)/2;
    tm = LoadBMPFile(fname1, x, y, 100);
    if (tm!=1){
        CloseGScreen();
        fprintf(stderr, "画像の読み込みに失敗しました。¥n");
        return(-1);
    }
    for (i=0;i<PNUM;i++){
        pagepal1[i].r = pal[i].r;
        pagepal1[i].g = pal[i].g;
        pagepal1[i].b = pal[i].b;
    }
```

```
/* 表示画像の切り替え */
    while ((c = getch()) != ESCAPE){

        switch(c){
            case UP:
                    page = (page + 1) % 2;
                    break;
            case DOWN:
                    page = (page + 1) % 2;
                    break;
            default:
                   continue;
        }
        switch(page){
            case 0:
#ifdef DOSV
                    SetVPage(3);
#endif
                    for (i=0;i<PNUM;i++){
                        SetBMPPalette(i, pagepal1[i].r, pagepal1[i].g,
                                    pagepal1[i].b, 100);
                    }
                    break;

            case 1:
#ifdef DOSV
                    SetVPage(3);
#endif
                    for (i=0;i<PNUM;i++){
                        SetBMPPalette(i, pagepal2[i].r, pagepal2[i].g,
                                    pagepal2[i].b, 100);
                    }
                    break;

       }
        WaitVSync();
        SetVPage(page);

    }

    CloseGScreen();
    return(1);

}
```