

EX-System

Books

Oh!X編集部編

X68000用グラフィックツールEX-System
EX-WIN, Z's-EX, Matier-EX & 各種画像フィルタ
各種関連ツール & 豪華なおまけ
綴じ込み付録 5"FD2枚組+CD-ROM

SOFT
BANK



ISBN4-89052-879-2

C0055 P4800E



9784890528790

定価4,800円

(本体4,660円)



1910055048003

主な収録ファイル一覧

EX-WIN.X グラフィックエディタ
Zs-EX.X Z'sSTAFF PRO-68K拡張プログラム
MAT-EX.X MATIER拡張プログラム
各種画像フィルタ+素材データ集
[おまけ]
Oh!X掲載プログラム
●SION IV完成版
●SX-BASICリファレンスマニュアル
●その他掲載プログラム
DoGA CGAシステム ver.ZZ
DoGA CGAシステム用背景データ集
Z-MUSICシステム ver.2.07+最新PCMデータ
Z-MUSICシステム ver.3お試し版
3D Atrier体験版(提供: Micronet)
Windows版EX-System(暫定版)

EX-System

Books

Oh!X編集部編

X68000用グラフィックツールEX-System
EX-WIN,Z's-EX,Matier-EX&各種画像フィルタ
各種関連ツール&豪華なおまけ
綴じ込み付録 5"FD2枚組+CD-ROM

SOFT
BANK



各種フィルタとツールの効果

EX-System にはさまざまな画像フィルタと特殊な効果を出すツールが揃えられています。ここではそれらの効果を具体的にまとめて見てみましょう。



▲元画像



▲ぼかし



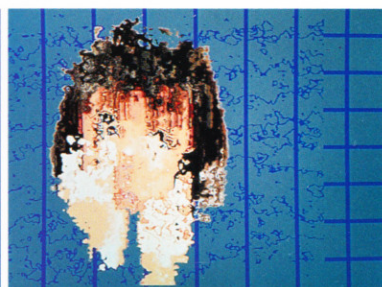
▲ぼかし（ディザつき）



▲横波変形



▲横波変形 2



▲ランダムフラクタル



▲色相フラクタル



▲フレア



▲トゥインクル



▲微分処理



▲合成



▲色反転

EX-System



▲アクセント (色強調)



▲サイクル



▲2値化



▲2値化 (ディザ)



▲ガラス化



▲ソフトフォーカス



▲ソフトフォーカス (4方向)



▲ソフトフォーカス (6方向)



▲色相回転



▲モーションブラー (接近)



▲モーションブラー (回転)



▲モーションブラー (平行)



▲主線合成



▲放射光



▲放射光 (彩色)

EX-System



▲パースペクティブ



▲円グラデ



▲PIC ペイント



▲陰影楕円



▲回転



▲回転 2



▲質感合成



▲球状変形



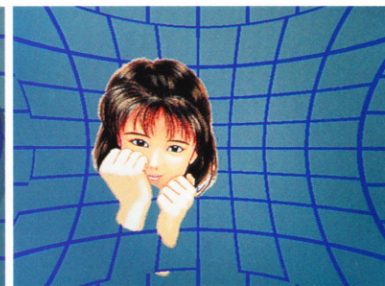
▲円錐変形



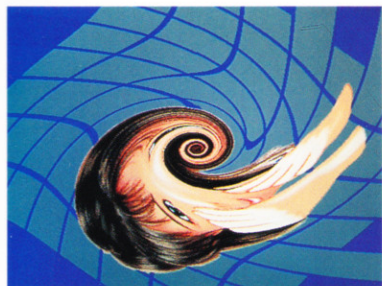
▲環管変形



▲放物面変形



▲収差変形



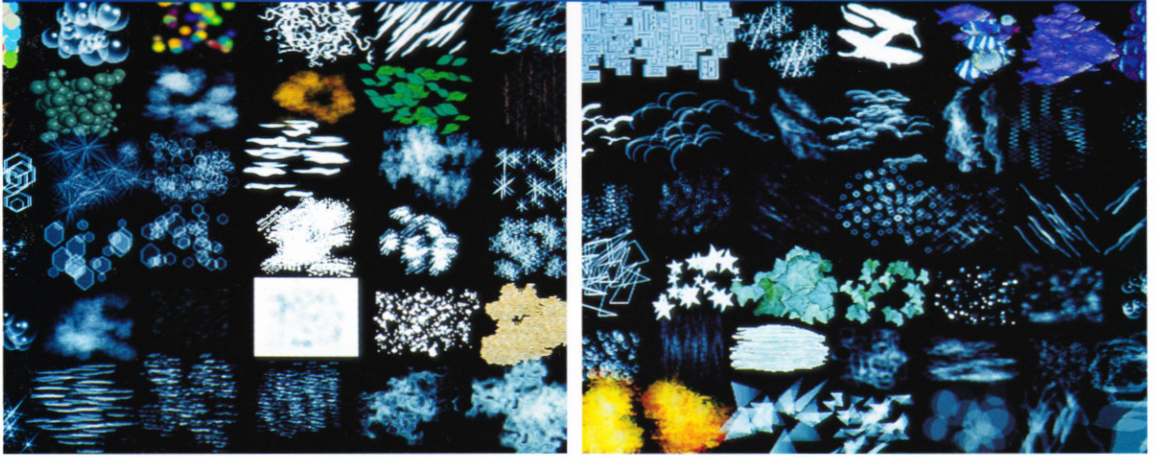
▲渦巻き



▲うのようによ



▲パターンブラシ



登録されているパターンブラシ一覧。各自で追加登録可能

雲の描き方



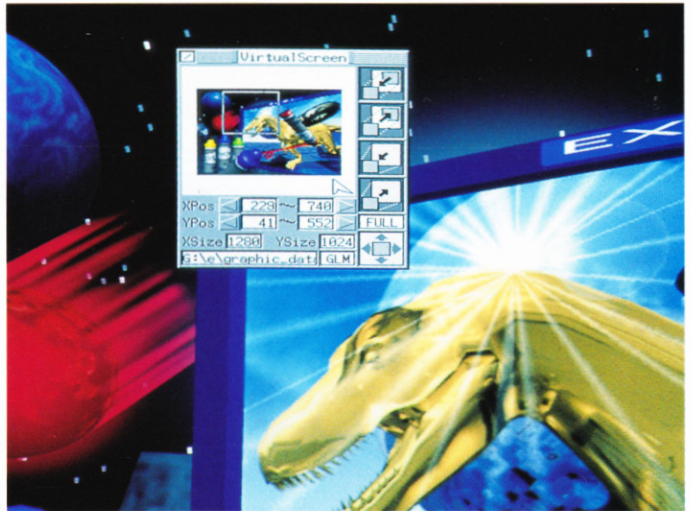
パターンブラシで模様を散らす



回転つきモーションブラーをかける



適当にパターンブラシで加筆する



画面に収まりきらない大きな画像は仮想画面によって部分ごとにエディットすることができる

CD-ROM に収録されている川原由唯氏のサンプル画像例





EX-System

CONTENTS

1 カラーギャラリー

6 第1章 イントロダクション

- 1. 1. はじめに
- 1. 2. EX-Systemの生まれるまで
- 1. 3. EX-System概要
- 1. 4. インストール

8 第2章 EX-Systemの使用法

- 2. 1. EX-Systemの違い
- 2. 2. 起動前の設定
- 2. 3. コンフィグファイル
- 2. 4. 起動
- 2. 4. 1. Z's-EXの起動
- 2. 4. 2. Matier-EXの起動
- 2. 5. 使用法
- 2. 5. 1. ファイルコントロール
- 2. 5. 2. パレットコントロール
- 2. 5. 3. ペンコントロール
- 2. 5. 4. エディットコントロール
- 2. 5. 5. ルーペコントロール
- 2. 5. 6. プラシコントロール
- 2. 5. 7. スタンプコントロール
- 2. 5. 8. フォントコントロール
- 2. 5. 9. スーパールーペコントロール
- 2. 5. 10. カスタムコントロール
- 2. 5. 11. 特殊なコントロール
- 2. 6. 便利な操作法
- 2. 7. 注意
- 2. 7. 1. Z's-EXでの注意
- 2. 7. 2. Matier-EXでの注意

19 第3章 付属外部ファイルの機能

- 3. 1. ルート階層
- 3. 1. 1. バージョン
- 3. 1. 2. 初期化
- 3. 1. 3. アスペクト比
- 3. 1. 4. 実行
- 3. 1. 5. 外部ファイラー
- 3. 1. 6. 表裏切り換え
- 3. 2. フィルタ階層
- 3. 2. 1. モノトーン
- 3. 2. 2. セピア
- 3. 2. 3. ランダムフラクタル
- 3. 2. 4. 色相フラクタル
- 3. 2. 5. フレア
- 3. 2. 6. フレア2
- 3. 2. 7. フレア3

- 3. 2. 8. 差分処理
- 3. 2. 9. 異相合成
- 3. 2. 10. 色強調
- 3. 2. 11. 色成分交換
- 3. 2. 12. ジャギー消去
- 3. 2. 13. 二値化
- 3. 2. 14. 二値化 (Dither)
- 3. 2. 15. ガラス化
- 3. 2. 16. ソフトフォーカス
- 3. 2. 17. クリスタルフレア
- 3. 2. 18. クリスタルフレア (hex)
- 3. 2. 19. ぼかし
- 3. 2. 20. ぼかし (Dither)
- 3. 2. 21. 色相変換
- 3. 2. 22. モーションブラー (接近)
- 3. 2. 23. モーションブラー (回転)
- 3. 2. 24. モーションブラー (平行)
- 3. 2. 25. 主線合成
- 3. 2. 26. 放射光
- 3. 2. 27. 質感合成
- 3. 2. 28. 明度調整
- 3. 2. 29. 明度加減算
- 3. 2. 30. Tilt
- 3. 2. 31. ういようによ
- 3. 2. 32. 収差変形
- 3. 2. 33. 鏡渡
- 3. 2. 34. 鏡渡 2
- 3. 3. ツール階層
- 3. 3. 1. 画面画パース変形
- 3. 3. 2. スクロー
- 3. 3. 3. パレット
- 3. 3. 4. 曲グラデーション
- 3. 3. 5. 拡大・縮小
- 3. 3. 6. プレーン
- 3. 3. 7. PICペイント
- 3. 3. 8. 回転
- 3. 3. 9. M/R
- 3. 3. 10. 陰影階層
- 3. 3. 11. バターンブラシ
- 3. 4. マスク階層
- 3. 4. 1. マスク操作
- 3. 4. 2. 線→マスク
- 3. 4. 3. マジックウィンド
- 3. 4. 4. ぼかし
- 3. 4. 5. 露取り
- 3. 5. 3次元階層
- 3. 5. 1. ライティング設定
- 3. 5. 2. 球状変形

- 3. 5. 3. 円錐変形
- 3. 5. 4. 環面変形
- 3. 5. 5. 放物変形
- 3. 5. 6. スクリプト
- 3. 6. サンプル階層

39 第4章 外部ファイルの作成

- 4. 1. 環境設定
- 4. 2. 外部ファイルへの引数と置き値
- 4. 3. 画面の構成
- 4. 4. 画面フィルタの作成
- 4. 4. 1. メディアンフィルタ
- 4. 4. 2. 減算加算
- 4. 4. 3. 画→表転送
- 4. 5. ウィンドウを開く外部ファイルの作成
- 4. 5. 2. ルーペ
- 4. 5. 3. テキストビュー
- 4. 6. システム画面ルーチンの利用
- 4. 6. 1. カレントドロ

47 第5章 とにかく使ってみよう

- 5. 1. ペンによる描画
- 5. 2. 合成の基本
- 5. 3. 筆を置く
- 5. 4. 自分のパターンブラシを作る

50 第6章 EX用ウィンドウデザイン

- 6. 1. コマンド
- 6. 2. コントロールアイテム
- 6. 3. C言語への変換
- 6. 4. 美しいユーザーインタフェイスを

54 第7章 AMIシステム

- 7. 1. SCSIとは
- 7. 2. SCSI用I/Oコール
- 7. 3. アニメーションのデータ構造
- 7. 4. 再生モードと再生能力
- 7. 5. AMIで使用するデータファイル
- 7. 6. 実行ファイル
- 7. 7. データの制作環境
- 7. 8. DoXデータのコンバート
- 7. 9. 小物ツール
- 7. 10. AMIシステム用ツール開発環境
- 7. 11. AMIファイルの構造

59 付録 EX-Systemコール一覧

1. イントロダクション

菊地 功

1. 1. はじめに

コンピュータの用途は人それぞれです。ゲームをしたり、コンピュータミュージックを楽しんだり、あるいは仕事で使ったり……。そんななかで、コンピュータグラフィック(CG)というのは、もっとも華やかなもののひとつでしょう。CGアーティストなんて職業が比較的舞台に出るようになり、一昔前なら「おっ、こりゃ凄い」といっていたようなCG(アニメーション)が、最近ではテレビ番組のオープニングなどで当たり前のよう使用されていたりします。

いまでこそ、パソコンでも発色数が大きく取り上げられるものは、それはつい最近になってからのことです。ちょっと前までは「パソコンに多色は贅沢だ」といったような風潮があり、とかく馬鹿にされがちでした。そんななかでX68000は、512KバイトのビデオRAMを引っ提げて登場したのですが、やはり当初は「あれは高価なゲーム機だ」という罵声があったようです。しかし、実際に触れてみれば、MC68000CPU、65536色同時表示、AD PCMなど、まさにパーソナルワークステーションの名にふさわしいマシンであったことが実感できたはずで

す。X68000の登場から8年が経ち、CPUは10MHzから16MHzへ、さらに68030へと変化しましたが、それ以外の基本的なスペックはなんら変わっていません。これを否定要因と見る人もいますが、裏を返せば8年前に現在でも通用するマシンを開発できたというのは、驚異といっても過言ではないでしょう。

さて、その発売当初から変わらない基本スペックのひとつ、65536色同時表示という機能により、X680x0にはそれを利用した多くのグラフィックツールが登場しました。現在その双璧となっているのが、ツァイトのZ's STAFF PRO-68K(以下Z's STAFFと略す)と、サンワードのMatierでしょう。Z's STAFFは、かなり初期のころに発売され、多くの人々に利用されながら、現在ではバージョン3.0に至っています。また、Matierは比較的最近の発売にもかかわらず、その充実した機能により、こちらも多くの人に愛用されています。どちらも優れたグラフィックツールなのですが、お互いに相手にはない長所を持ち合わせており、むしろ機能よりも趣味の問題で意見が大きく分かれるところでしょう。そんななかで、新たなグラフィックツール、EX-Systemの登場です。

1. 2. EX-Systemの生まれるまで

EX-Systemの起源は、満開製作所の月刊電腦倶楽部に掲載されたPICFILERというツールです。これは、普段はZ's STAFFの裏に隠れているのですが、キーボードから「S」キーを押すことで呼び出され、X680x0の画像フォーマットとして揺るぎない地位を得たPICファイルを、Z's STAFFからロードおよびセーブできるようにするものです。

さらに、昔からのOh!Xの読者ならばご存じだと思いますが、そのPICFILERを拡張し、いくつかの特殊効果を付け加えたものが、初代Z's-EXです。これは丹明彦氏により作成され、Oh!Xの1991年1月号に掲載されました。これにより、Z's STAFFは新たな機能を付加されることになったのですが、それでもZ's-EXが提供する機能が追加されるのみで、閉じた環境でしかありませんでした。

そこで、もっと開いた環境をと、御木徳高氏によって改良され、Oh!X1992年2月号に掲載されたのがZ's-EX Ver. 1.1です。これは、任意の実行形式のコマンドを(外部ファイルと呼ぶことにします)Z's STAFFから呼び出すことができるようにするものでした。これにより、あらかじめ登録しておけば、ユーザーがある規則に則って作成した外部ファイルを自由に実行できるようになったのです。しかし、これは本当にただ実行できるだけであって、システムと外部ファイル間での情報のやりとりがほとんど行えなかったで、実際問題としてせいぜいフィルタ程度の外部ファイルしか作成できなかったのです。

今度は外部ファイルとのインタフェースを主眼に入れ、改良を行ったのがZ's-EX ver. 2.0です。これはOh!X1994年3月号に掲載されました。外部ファイルはファンクションコール(EXコールと呼ぶことにします)を使用して、システムの情報を得たり、機能を利用することができるのです。このEXコールにより、外部ファイル側から簡単にウィンドウの作成などもできるようになりました。また、これと同時にMatierの機能を拡張するMatier-EXも発表され、直後に単独で駆動するEX-Windowが発表されました。これら3つのEXを総称して「EX-System」と呼びます。

そして、今回のEX-System ver. 3.0です。これはZ's-EX ver. 3.0、Matier-EX ver. 3.0およびEX-Window ver. 3.0を指します(注1)。いまでは、基本的に外部ファイルを実行するためのプラットフォームとしての機能しか持ちあわせておらず、単独ではメモリを圧迫するだけの代物だったのですが、ver. 3.0ではそれ自身が描画機能を有するようになり、グラフィックツールとして生まれ変わりました。EXコールもそれに伴って強化され、より使いやすくするために、細かい機能付加も行われましたが、一部を除いて(注2)上位互換を保っていますので、以前のバージョンのほとんどの外部ファイルをそのまま使用することができます。

(注1) Matier-EX ver. 1.0はMatier ver. 1.0専用、Matier-EX ver. 2.0はMatier ver. 2.0専用でしたが、Matier-EX ver. 3.0はMatier ver. 1.0、2.0両用です。また、EX-Windowは、ほかとバージョンを合わせるために、ver. 2.0をスキップしました。

(注2) アナログマスクのレベルが若干変更されました。詳しくは第4章「外部ファイルの作成」を参照してください。

1. 3. EX-System概要

EX-Systemはグラフィックツールです。Z's-EXおよびMatier-EXは、それぞれZ's STAFFおよびMatierを内部から呼び出し、起動時には表に現れません。その後、シグナル(注3)を受け取ることで、EX-Systemが起動し、Z's-EXやMatier-EXの機能拡張を行うことができます。ま

た、EX-Windowは、単独で駆動しますので、Z's-EXやMatier-EXと比べて、少ないメモリでEX-Systemの機能を利用できます。

EX-Systemは、さまざまな機能を提供するために、かなりのメインメモリを必要とします。フリーエリアが少ないときには、いくつかの機能を切り離して起動することも可能ですが、動作に制限ができますので、メモリを増設するなどして、ぜひともフルスペックのEX-Systemを体験してみてください。きっと納得していただけるはずです。EX-Windowならばフリーエリアが4Mバイト以上あれば完全に動作します（Z's-EXやMatier-EXでは、プラス1～2Mバイト）。さらに、場合によってはCPUパワー、もしくは忍耐を必要とします。ご了承ください（機能に変化はありません）。

EX-Systemは、画面に表示されている作業画面のほか、デフォルトでもう一枚裏画面を持っています。この裏画面の一部を作業画面にコピーしたり、裏画面の色で作業画面にペイント、ドロウなどを施したりするだけでなく、外部ファイルによってさまざまな用途で使われます。

Z's STAFFやMatierでは、マスクは1ビット、すなわちONかOFFでしたが、EX-Systemでは、マスクを8ビットに拡張し（実際に使用しているのは5ビット）、32段階のアナログマスクを実現します。これにより、マスクの上からの描画は、そのマスクのレベルによってマスク下に透過され、原画像に合成されます。また、外部ファイルによって扱いが多少変わる場合もあります（無視されることもあります）。

この2つの機能に関しては、起動時のオプションで切り離すことができます（オプションについては、第2章「EX-Systemの使用法」を参照してください）。アナログマスク、裏画面の順で切り離され、前者と切り離したものをスモールモデル、両者を切り離したものをタイニーモデルと呼びます。

コンフィグファイルにあらかじめ特定のフォーマットで登録しておくことで、任意の外部ファイルを実行することができます。外部ファイルは階層状に登録できますので、関係の深いものをまとめておけば混乱を防ぐことができます。外部ファイルは、C言語用のライブラリと、アセンブラ用のマクロを用意しており、吉田泉氏によるウィンドウデザインも付属していますので、比較的簡単に作成することができます。

（注3）EX-Systemの起動方法は、Z's-EXとMatier-EXで異なります。詳しくは第2章「EX-Systemの使用法」を参照してください。

1. 4. インストール

インストールには若干、Human68kの知識が必要となります。「ファイルのコピー」「パス（環境変数）の設定」「テキストエディタの使い方」などは、Human68kのマニュアルを参照してください。

EX-Systemは、ハードディスクにインストールすることを前提にしています。フロッピーディスクでの運用も不可能ではありませんが、頻繁にディスクにアクセスにいきますし、システムと外部ファイルのすべてを2枚のディスクに収めることはできませんので、ここではハードディスクへのインストールの解説をします。どうしてもフロッピー

ーディスクで運用したいという方は、個人の責任において行ってください。

EX-Systemは、主に3種類の実行ファイルと、それに付随するデータファイルで構成されています。まずひとつはEX-Systemの本体、EX_Win. X、Zs_EX. XおよびMat_EX. Xの3ファイル（それぞれEX-Window、Z's-EX、Matier-EX本体）と、それに付随するEX_Win. SYS、Zs_EX. X. SYS、Mat_EX. SYSとEX_Icon. DATです。これらはパスの通ったディレクトリに置かなければなりません、3つの本体を違うディレクトリに置いて構いません。ただし、EX_Win. SYSはEX_Win. Xと、Zs_EX. SYSはZs_EX. Xと、Mat_EX. SYSはMat_EX. Xと同じディレクトリに置かなければならず、EX_Icon. DATはすべてが必要とするので、それぞれのディレクトリに置かなければなりません。

2つ目は、EX-Systemが直接呼び出すシステムファイルです。各種コントロールウィンドウや、描画関係のルーチンで、ユーザーが呼び出すことは禁止されています。表1.1にそれらのファイルを示します。それぞれの実行形式のファイルは、パスが通っていればどこに置いて構いませんが、それぞれに付随するファイルは同じディレクトリに置かなければなりません。

3つ目は、外部ファイル群です。これもパスが通っていればどこに置いて構いません。これらの機能や、付随するファイルなどは、第3章「付属外部ファイルの機能」を参照してください。

ファイルのコピーが終了したら、いくつかのファイルを自分の環境にあわせて、テキストエディタなどで書き換えてやる必要があります。システムが使用するファイルに関しては、第2章「EX-Systemの使用法」で、外部ファイルが使用するファイルに関しては、第3章「付属外部ファイルの機能」で解説します。

表1.1 EX-System本体が呼び出すシステムファイル一覧

| ファイル名 | 機能 | 付随データファイル |
|---------------|----------------|---------------------------|
| EXConfig.x | システム設定 | |
| EXType.x | コンソールログ表示 | |
| EX_File.x | ファイルメニュー | |
| EX_Pal.x | パレットメニュー | |
| EX_Pen.x | ペンメニュー | EX_Pen.DAT、EX_PenIcon.DAT |
| EX_Edit.x | エディットメニュー | EX_EditIcon.DAT |
| EX_Lupe.x | ルーペメニュー | |
| EX_Brush.x | ブラシメニュー | EX_BrushIcon.DAT、RF.DAT |
| EX_Stamp.x | スタンプメニュー | EX_Stamp.SYS |
| EX_Font.x | フォントメニュー | EX_Font.SYS |
| EX_Super.x | スーパールーペメニュー | EX_SuperIcon.DAT |
| EX_Custom.x | カスタムメニュー | |
| EXPoint.x | 自由曲線描画ルーチン | |
| EXLine.x | 直線描画ルーチン | |
| EXBox.x | 長方形描画ルーチン | |
| EXBoxFill.x | 塗りつぶし長方形描画ルーチン | |
| EXEllipse.x | 円描画ルーチン | |
| EXFill.x | 塗りつぶし円描画ルーチン | |
| EXPaint.x | 塗りつぶし描画ルーチン | |
| EXGradation.x | グラデーション描画ルーチン | |
| EXCopy.x | コピールーチン | |
| EXResize.x | リサイズコピールーチン | |
| EXReverse.x | 色反転ルーチン | |
| EXMirror.x | 左右反転ルーチン | |
| EXFlip.x | 上下反転ルーチン | |
| EXChange.x | 色変換ルーチン | |

このうち、EX_Stamp.SYSとEX_Font.SYSは使用環境にあわせて書き換える必要がある。詳しくは第2章「EX-Windowの使用法」を参照してください。

2. EX-Systemの使用法

菊地 功

2. 1. EX-Systemの違い

Z's-EXやMatier-EXは、通常のグラフィックツールとは違い、特定のグラフィックツールに寄生して必要に応じて呼び出されるという特殊な方式をとっています。これにより、既存の優れたグラフィックツールを終了することなく、2つのツールを行き来することができます。

Z's-EXやMatier-EXは、起動されると内部からZ's STAFFやMatierを呼び出し、自分は裏方にまわって起動シグナルを待つことは前章でお話ししました。そのシグナルとなるのが、Z's-EXでは、起動キー（デフォルトでは'S'キー）を押すこと、Matier-EXではMatierのコマンド実行機能を利用してEXOPEN.Xを実行することです。また、Z's STAFF ver. 3.0ではZ's STAFFのコマンド実行機能を利用してEXOPEN.Xを実行することでも、Z's-EXを呼び出すことができます。これにより、EX-Systemのメニューウィンドウが開きますが、EX-Windowでは、起動するとすぐにこの状態となり、以降はすべてのEX-Systemで共通の操作を行うことができます。

2. 2. 起動前の設定

ハードディスクに必要なファイルを移したあと、EX-Systemを起動する前には、いくつかのファイルを書き換えたり、環境変数などの設定変更が必要となります。

まず、EX-System本体、システムファイル、外部ファイルのあるディレクトリすべてにパスを通します。AUTO EXEC. BATの中や、起動用のバッチファイルを作って、設定するといいでしょ。

次にスタンプ設定ディレクトリファイルEX_Stamp.SYSと、フォント設定ファイルEX_Font.SYSなどの*.SYSとついたファイルをテキストエディタなどで各自の環境にあわせて書き換えます。

EX_Stamp.SYSには、スタンプデータを保存しておくディレクトリを記述します。混乱を避けるため、新たなディレクトリを作成し、そのディレクトリを指定することをおすすめします。

表2.1 コンフィグファイルのフォーマット

```
mem, key
:title1
    flag, pn[:p1min-p1max, p1default[:p2min-p2max, p2default]]
    filename1 [option]
    [filename2 [option]]
:title2
    flag, pn[:p1min-p1max, p1default[:p2min-p2max, p2default]]
    filename1 [option]
    [filename2 [option]]
:title3
{
    :title4
        flag, pn[:p1min-p1max, p1default
                [:p2min-p2max, p2 default]]
        filename1 [option]
        [filename2 [option]]
}
```

EX_Font.SYSには、登録したいフォントファイルをフルパスで、それぞれ改行で区切って記述します。登録できるフォントは、SX-WINDOW付属のIFMフォントと、書体倶楽部のフォントです。書体倶楽部のフォントは、拡張子がVF1のもの（JIS第一水準）だけを記述しておけば、VF2（JIS第二水準）の記述は必要ありません。ディスクにサンプルを添付しておきましたので、そちらも参考にしてください。これらのファイルの書き換えは初めて起動するときや環境が変わったときのみ行う必要があります。

また、新しく外部ファイルを登録するには、コンフィグファイルを書き換える必要がありますが、添付されているコンフィグファイルには、今回添付したすべての外部ファイルの登録をしてありますので、それ以外の外部ファイルを登録したい場合を除いて書き換える必要はありません。コンフィグファイルの設定については、後述します。

Z's-EXを起動するときには、Z's STAFF本体のSTAFF68K.Xのあるディレクトリにカレントを移しておかなければなりません。

Matier-EXを起動するときには、環境変数“MATIER”（すべて大文字半角）に、Matier本体のMAT.Xのあるディレクトリを設定します。これはMatierからも参照する環境変数ですから、Matierがインストールされていれば、すでに設定されているかもしれません。また、Matier ver. 1.0では、Matier定義ファイルMAT.DEFで、チャイルドプロセスを実行するためのメモリを確保してやらなければなりません。512Kバイト以上が推奨ですが、それ以下では容量に応じて実行できない外部ファイルが現れます。

MatierからMatier-EXを呼び出すシグナルとなるEXOPEN.Xは、一度COMMAND.Xを実行してコマンドラインから実行しても構わないのですが、MAT.DEFを書き換えてファンクションキーに登録しておけば、スムーズな行き来ができるようになります。EXOPEN.Xにはオプションがあり、1～4の整数値でMatierの何番目の裏画面をMatier-EXの裏画面にするかを指定できます（省略時は1、確保されていない裏画面を指定した場合は、「確保されていません」のメッセージが表示され、Matier-EXは呼び出されません）。F6～F9に、EXOPEN 1～4とでも登録しておくとう便利でしょう。

2. 3. コンフィグファイル

EX-Systemに外部ファイルを登録する際の設定ファイルを、コンフィグファイルといいます。デフォルトでは、EX-WindowのコンフィグファイルはEX_Win.SYS、Z's-EXはZs_EX.SYS、Matier-EXはMat_EX.SYSですが、起動時のオプションで指定することもできます（オプションは後述します）。

コンフィグファイルのフォーマットを、表2.1に示します。

mem

外部ファイル実行用メモリのバイト数を10進で記述します。Z's-EXでのみ有効で、Matier-EXやEX-Windowでは、記述しても無視されます。外部ファイルによって使用するメモリ量は違いますので、もっともメモリを多く必要とする外部ファイルにあわせて指定してください。

key

Z's-EXの起動キーを2桁の数字で記述します。1桁目

がキーコードグループ、2桁目がキービットを表します。表2.2にキー対応表を示しますので、参考にしてください。省略時の起動キーは37、すなわち‘S’キーになります。これもZ’s-EXでのみ有効で、Matier-EXやEX-Windowでは、記述しても無視されます。

title

機能名です。EX-Systemから外部ファイルを実行するときには、この機能名を選択することになります。文字数は半角で16文字までですが、パラメータの数によって、表示が衝突してしまうことがあります。

flag

外部ファイルを起動する前に、EX-Systemが行う処理を指定します。0のときはそのまま外部ファイルを実行しますが、1を指定すると、外部ファイル実行前に矩形指定を行ってから、その座標を外部ファイルに渡すようにします。

pn

EX-Systemでは外部ファイル実行時に1桁の整数値を最大2つまで、パラメータとして外部ファイルに渡すことができます。pnにはパラメータの数を0～2の範囲で指定します。

p1min, p1max, p1default

p2min, p2max, p2default

それぞれひとつ目、2つ目のパラメータの最小値、最大値、初期値で、パラメータの数にあわせて記述してください。最大値は最小値より大きくなければならず、初期値はその範囲内でなければなりません。すべて1桁の整数値で記述します。

filename

実際の外部ファイル名を指定します。ファイル名はベース名（パス名と拡張子を省いたもの）だけを指定してください。EX-Systemでは、ひとつの機能名で2つの外部ファイルを連続実行することができます。その場合、指定されれば矩形指定範囲、パラメータは双方の外部ファイルに渡されます。

option

外部ファイルごとに固定のオプションを指定します。オプションの指定は半角33文字までです。

また、EX-Systemは機能を階層状に整理することができます。階層の登録には、機能の登録と同様にタイトルを記述して、階層内に入れたい機能を‘{’と‘}’でくくってください。階層内にさらに階層を入れることもできます。

インデント（行先頭の桁合わせ）はしなくても構いませんが、改行位置は守ってください。矩形指定やパラメータ、オプションは、対応した外部ファイルに対してのみ有効です。今回収録した外部ファイルの対応は、第3章「付属外部ファイルの機能」を参照してください。

2. 4. 起動

EX-Systemは、起動時にオプションを指定することで、機能の切り離しなどの設定を行うことができます。オプション一覧を、表2.3に示します。

Z’s-EXで‘/n’オプションを指定すると、起動キーでZ’s-EXを呼び出せなくなります。その場合、Z’s-EXの呼び出しはZ’s STAFFのコマンド実行機能でEXOPEN.

表2.2 キーコードグループ(縦)とビット(横)の対応

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|--------|--------|------|------|------|------|-------|------|
| 0 | | ESC | 1 | 2 * | 3 # | 4 \$ | 5 % | 6 & |
| 1 | 7 ‘ | 8 (| 9) | 0 - | = | ^ | ¥ | BS |
| 2 | TAB | Q | W | E | R | T | Y | U |
| 3 | I | O | P | @ | [| CR | A | S |
| 4 | D | F | G | H | J | K | L | : |
| 5 | : |] Z | X | C | V | B | N | . |
| 6 | M | . < | . > | / ? | — | SP | HOME | DEL |
| 7 | RollUP | RollOW | UNDO | ← | ↑ | → | ↓ | CLR |
| 8 | / | * | - | 7 | 8 | 9 | + | 4 |
| 9 | 5 | 6 | = | 1 | 2 | 3 | ENTER | 0 |
| A | . | . | 記号 | 登録 | HELP | XF 1 | XF 2 | XF 3 |
| B | XF 4 | XF 5 | かな | 0-2字 | 3-1* | CAPS | INS | 05桁 |
| C | 全角 | BREAK | COPY | F 1 | F 2 | F 3 | F 4 | F 5 |
| D | F 6 | F 7 | F 8 | F 9 | F 10 | | | |
| E | SHIFT | CTRL | OPT1 | OPT2 | | | | |
| F | | | | | | | | |

表2.3 EX-Systemのオプション一覧

| EX-Windowのオプション | |
|-----------------|---|
| /t | タイニーモデルを起動します |
| /s | スモールモデルを起動します |
| /f(filename) | コンフィグファイルを指定します |
| /u | アンドゥバッファを確保しません |
| Z’s-EXのオプション | |
| /t | タイニーモデルを起動します |
| /s | スモールモデルを起動します |
| /f(filename) | コンフィグファイルを指定します |
| /n | キー割り込み起動しません |
| /u | Z’s STAFF PRO68K ver3.0以降で使用 アンドゥバッファを確保しません |
| Matier-EXのオプション | |
| /s | スモールモデルを起動します |
| /f(filename) | コンフィグファイルを指定します |
| /u | アンドゥバッファを確保しません |
| <Others> | MATIERに渡します |

Xを実行することで行います。

Matier-EXは、Matierの裏画面を使用することができますので、タイニーモデルは存在しません（Matierで裏画面は必ず確保しなければなりません）。

オプションを指定しなくても、起動時にメモリが不足していた場合には、アンドゥバッファ、アナログマスク、裏画面の順で切り離されます。アンドゥとは、ひとつ前の操作をキャンセルし、その前の状態に戻る機能ですが、アンドゥバッファを確保しなかった場合には、アンドゥは使用できません。

2. 4. 1. Z’s-EXの起動

Z’s-EXを起動すると、まずZ’s STAFFのいつもの画面が現れるはずですが、そうならなかった場合は以下の点を確認してください。

・Z’s STAFFが起動されなかった場合

カレントがSTAFF68K.Xのあるディレクトリになっていませんか？ Z’s-EXが必要とするファイルが、Zs_EX.Xと同じディレクトリにありますか？ メモリは足りていますか？ メモリが足りない場合は、デバイスドライバ、常

駐物、RAMディスクを極力外してからもう一度実行してください。それでもだめな場合は、残念ながらその環境ではZ's-EXは使用できません。メモリを増設してください。

また、コンフィグファイルの記述が間違っている場合には、そのまま黙ってしまうことがあります。いったんインタラプトスイッチを押してコマンドラインに戻ってから、コンフィグファイルをチェックしてください。

・画面が真っ白になって動かなくなる

Z's STAFFが、メモリの確保に失敗しているようです。いったんリセットしてから（たぶんインタラプトでは正常に戻りません）、オプションで機能を切り離すか、コンフィグファイル中で指定している外部ファイル実行用メモリを小さく、あるいは思い切って大きくしてください。

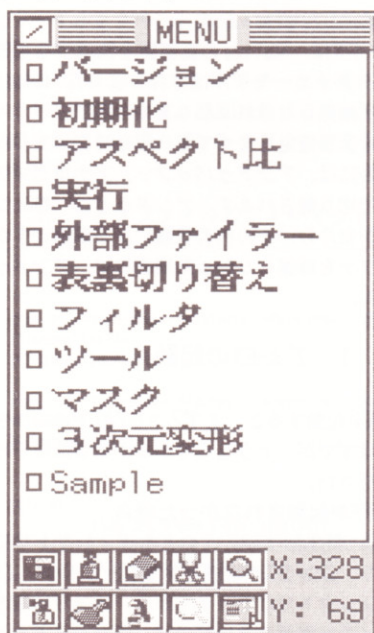
・Z's STAFFのマウスカースル/アイコンが表示されない

カレントディレクトリがSTAFF68K.Xのあるディレクトリになっていますか？ STAFF68K.Xが必要とするファイルが同じディレクトリにありますか？

これら以外の症状でも、デバイスドライバがぶつかっている場合などが考えられます。また、一見正常に動作しているように見えても、終了すると割り込みが狂っている場合がありますので、起動前にはできるだけ常駐物を外しておいてください。

無事にZ's STAFFが起動したら、次にZ's-EXを呼び出してみましょう。起動キーを押すだけですが、その前にウィンドウが開いていたら、すべて閉じておいてください（メニューバーは表示したままでも構いません）。これは必ず守ってもらわなければならない、ウィンドウが表示されたままZ's-EXを起動した場合の動作は保証できません。コンフィグファイルで起動キーを変更していなければ、'S'キーでZ's-EXが呼び出されます。あるいはZ's STAFF ver. 3.0からならEXOPEN.Xを実行することでも呼び出すことができますが、その場合、オプションに '+' をつけることでウィンドウを閉じる手間が省けます（ '+' をつけない場合はタイトルバーも消しておく必要があります）。

図2.1 メニューウィンドウ



2. 4. 2. Matier-EXの起動

Matier-EXを起動すると、まずMatierのいつもの画面が現れます。Matier-EXの起動は、ほとんど問題が起こることはないと思いますが、あるとすればパスが通っていない、必要なファイルがないといったことくらいでしょう。

Matier-EXの呼び出し方は、MAT.DEFの設定を変えていなければ、F1でコマンドラインに降りてから、

EXOPEN <リターン>

と入力するか、ファンクションキーに登録してあれば、登録したキーを押すだけです。このとき、EXOPENの引数として裏画面番号を渡すときには、Matier側で必ずそれに対応した裏画面を確保してある必要があります。同じ領域を使用していますので、Matier-EXで裏画面を操作した場合には、Matierの裏画面も更新されます。また、Matier-EXは、Matierのアンドゥバッファを表画面として使用しますので、Matier-EXを呼び出すとMatierのアンドゥでそれ以前の状態に戻すことはできません。

2. 5. 使用法

ここでは、EX-Systemのシステム本体の使用法について解説します。EX-Systemは、シングルウィンドウシステムで、一部の操作を除いてマウスオペレーションとなっています。多くのウィンドウは、いちばん上の部分がタイトルバーとなっており、そこをドラッグ（マウスボタンを押しながらマウスを移動させる）すると、ウィンドウを移動させることができます。また、システムのウィンドウのいちばん下には、10個のアイコンが並べられています。これらはクリックすることで、後述のコントロールウィンドウを表示することができ、F1~F10キーを押すことでも同様の動作をします。

EX-Systemが起動されると、まずメニューウィンドウがオープンします（図2.1）。このウィンドウは外部ファイルを実行するためのウィンドウで、ウィンドウの中にはコンフィグファイルで登録された機能名と階層名が表示されています。それぞれの機能を実行、または階層に降りる場合は、それらのいちばん左の四角いボタンにマウスカースルをあわせて、マウスボタンをクリックしてください。パラメータがある場合には、パラメータボックスが右端に表示され、そこをマウスで左右クリックすることで、パラメータの値を加減できます。また、機能名および階層名がたくさんあり、ウィンドウ内に表示しきれない場合は、ウィンドウの中央あたりでマウスを左右クリックすることで、上下スクロールさせることができます。ウィンドウ左上のクローズボックスは、クリックすることで親階層に抜けることができ、いちばん上の階層ではEX-Systemを終了することができます（Z's-EXとMatier-EXでは、それぞれZ's STAFFとMatierに戻ります）。ウィンドウの中央で左右ボタンを同時クリックすることでも同様の働きをします。また、ESCキーを押すと無条件にEX-Systemを終了できます（Z's-EXとMatier-EXでは、次に呼び出されたときは最後にいた階層が開きます）。ウィンドウの外での左クリックで、カレントドロモードでの描画/編集を行い、右クリックでスポイトとなり、カレントカラーまたはマスクにその位置のカラーまたはマスクがセットされます。

EX-Systemでは、以下のようなキーボードからの操作により、特殊な操作を行うことができます。

- ・スペースキー

裏画面が使用できる場合は、作業画面と裏画面を交換します。

- ・XF1キー

押しているあいだけマスクを透視することができます。

- ・XF3キー

作業画面のマスクのレベルを反転します。

- ・XF4キー

裏画面が使用できる場合は作業画面と裏画面のマスクを交換します。

- ・SHIFT+XF5キー

作業画面のマスクをクリアします。

- ・UNDOキー

アンドゥバッファがある場合はアンドゥを行います。

- ・登録キー

アンドゥバッファが確保されている場合は、作業画面をアンドゥバッファに転送します（フィックス）。

- ・CLRキー

画面のドットアスペクト比を4:3←→1:1にトグル切り換えます（Z's-EXでは機能しません）。

通常、フィックスが行われるのは、描画/編集モードが変更されたとき、ブラシ、スタンプ、フォントコントロールを開いたとき、スーパールーペでなんらかの操作を行ったとき、および外部ファイルを実行したときです。スペースキーにより作業画面と裏画面を交換した場合は、フィックスは行われませんので、注意してください。また、後述のシステムコントロールでオートフィックスを無効にした場合、またはCTRLキーを押しながら上の操作を行った場合には、フィックスを行うことをしません。

次に、コントロールウィンドウについて解説します。それぞれのコントロール内では、上で示したキー操作のほかに、XF2キーでウィンドウの表示/非表示を切り換えることができます。

2. 5. 1. ファイルコントロール

フロッピーディスクのアイコン、もしくはF1キーを押すことでオープンされます（図2.2）。ここでは、画像およびマスクのロード/セーブを行うことができます。画像はPIC形式のもののみを扱うことができます。

- ・ドライブアイコン

カレントドライブを示します。マウスでクリックすることで、ドライブを移動させることができます。

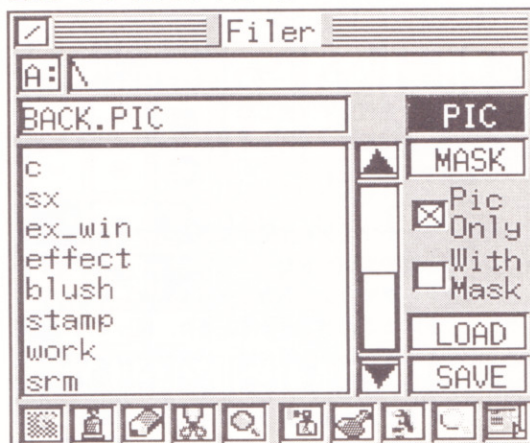
- ・ディレクトリボックス

カレントディレクトリを示します。マウスでクリックして、キーボードからディレクトリ名を入力することで、カレントディレクトリを変更することができます。シングルクリックで編集、ダブルクリックで新規入力となり、カレントディレクトリからの相対パス指定（“.”、“..”は不可）や、ドライブ指定もできます。

- ・ファイルボックス

カレントファイルを示します。ロード/セーブは常にここに示されたファイル名で行われます。ディレクトリボックス同様、相対パス指定やドライブ指定もできます。なお、拡張子は自動的に付加されます。

図2.2 ファイルコントロール



- ・ファイルメニュー

カレントディレクトリにある指定ファイルおよびディレクトリを示します。ファイル名は黒で、ディレクトリ名は灰色で表示されます。

ファイル名をクリックした場合は、ファイル名がファイルボックスに転送され、ディレクトリの場合は、そのディレクトリにカレントが移動します。また、ファイル名をダブルクリックした場合は、そのファイルのロードを行います。

- ・スクロールアイコン

ファイル名およびディレクトリ名がファイルメニューに収まらないときに、スクロールアイコン部分をクリックすることで、ファイルメニューをスクロールさせることができます。

- ・スクロールバー

カレントディレクトリにあるファイル数およびディレクトリの量や、ファイルメニューの相対位置の目安を知ることができます。また、この部分を直接クリックすることでファイルメニューを対応する位置まで移動させることができます。

- ・PICアイコン

画像のロードおよびセーブを行うときには、このアイコンをクリックして反転させ、PICモードにします。黒地に白文字が動作モードです。

- ・MASKアイコン

マスクのロードおよびセーブを行うときには、このアイコンをクリックして反転させ、MASKモードにします。黒地に白文字が動作モードです。

- ・PICオプション

PICモードで、マスクと一緒にロード/セーブを行うかどうかを指定します。“With Maskが”チェックされているときは、PICモードのカレントディレクトリに、ファイルボックスで示されているファイル名の拡張子を“.MSK”に変換したファイル名で、マスクのロード/セーブを行います。

- ・LOADアイコン

カレントディレクトリから、ファイルボックスで示されたファイルをロードします。

- ・SAVEアイコン

カレントディレクトリに、ファイルボックスで示されたファイルをセーブします。

2. 5. 2. パレットコントロール

絵の具のアイコンをクリック、もしくはF2キーを押すことでオープンされます(図2.3)。

ここでは、主に描画ソースとなるカラーやマスクを操作することができます。

・パレット

16個のカラーやマスクを保持することができます。右または左クリックでカレントカラー(マスク)へのゲット、左右同時クリックでカレントカラー(マスク)をそのブロックにセットすることができます。また、誤ってパレットに色をセットしてしまった場合は、パレットコントロールを抜けない限り、CTRLキーを押しながら特定のパレットをクリックすることで前のカラー(マスク)に戻すことができます。

・レベルバー

カレントカラーおよびカレントマスクの各要素のレベルを示します。バーをドラッグしたり、レベルバー右のボックスをクリックして、キーボードから数値を入力することで値を変更することができます。また、バー左のラベルを左右クリックすることでも、レベルを増減することができます。マスクのレベルは、大きいほど透過率が大きくなり、0で不透明、31で透明(マスクのない状態)となります。レベル31のマスクは作業画面では表示されませんが、各メニューでは緑の点滅で表されることがあります。

・カレントカラー

カレントカラーを示します。描画ソースがカレントカラーのときに使用される色です。

・カレントマスク

カレントマスクを示します。描画ソースがカレントマスクまたはリムーブマスクのときに使用されるマスクです。

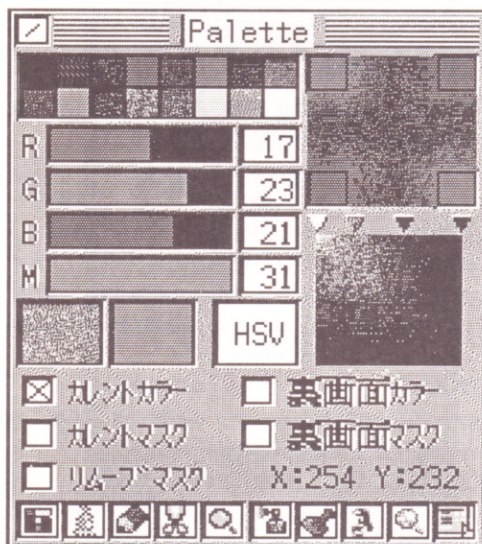
・RGB-HSV切り換えボタン

レベルバーの、カレントカラーの要素であるRGBとHSVを切り換えます。

・描画ソース

描画時の描画元になるものを示します。一部では利のないものもあります。

図2.3 パレットコントロール



1) カレントカラー

カレントカラーボックスに示されたカラーで、描画が行われます。

2) カレントマスク

カレントマスクボックスに示されたマスクが、描画が行われます。

3) リムーブマスク

カレントマスクのレベルに応じて、マスクを薄く(透過する方向)させます。レベルが小さいほど薄くする度合いが強くなり、0で完全にマスクを除去します。

4) 裏画面カラー

裏画面が使用できる環境ならば、対象となる作業画面と一致する座標の裏画面のカラーで描画されます。

5) 裏画面マスク

裏画面が使用できる環境ならば、対象となる作業画面と一致する座標の裏画面のマスクで描画されます。

スポイトやパレットのセット、後述のグラデーションの表示などは、この描画ソースがカレントカラーまたは裏画面カラーの場合はカラー、それ以外はマスクになります。

・平方グラデーション

四隅のボックスにカラー(マスク)をセットすることで、グラデーションを生成します。四隅のボックスを左クリックすることで、カレントカラー(マスク)をセットでき、右クリックでその位置のカラー(マスク)をカレントカラー(マスク)にゲットすることができます。生成したグラデーションは、グラデーションの任意の位置をクリックすることでスポイトすることができますし、後述の平方グラデーション描画で使用することもできます。

・中継グラデーション

グラデーションの上の4つの三角のボタンにカラー(マスク)をセットすることで、それぞれを経由したグラデーションを生成します。三角のボタンを右クリックすることでその位置のカラー(マスク)をカレントカラー(マスク)にゲット、左右同時クリックすることでカレントカラー(マスク)をその位置にセットすることができます。また、中間の2つのボタンはドラッグすることで位置を移動させることができます。生成したグラデーションは、グラデーションの任意の位置をクリックすることでスポイトすることができますし、後述の水平グラデーション描画などで使用することもできます。

裏画面が使用できる環境ならば、対象となる作業画面と一致する座標の裏画面のマスクで描画されます。

・ウィンドウ外

左クリックで、カレントドロモードでの描画/編集を行います。右クリックでスポイトとなり、カレントカラーまたはマスクにその位置のカラーまたはマスクがセットされます。

2. 5. 3. ペンコントロール

鉛筆のアイコンをクリック、もしくはF3キーを押すことでオープンされます(図2.4)。ここでは、描画に使用されるペンや、描画方法を決定することができます。

・ペンアイコン

カレントペンを決定するアイコンです。左上のアイコンはドットペンであり、その下の左の列の4つは4×4、中央の4つは8×8、右は16×16のパターンペンです。ドッ

トペンを除いて、ダブルクリックでペンエディットウィンドウが開き、パターンを編集することができます。

・ペン濃度

パターンペンの濃度を決定します。値が大きいくほど濃度は濃くなります。

・描画アイコン

カレントの描画方法を決定するアイコンです。

1) 自由曲線

マウスをドラッグすることで、指定された描画ソースとペンで自由曲線を描画します。

2) 直線

マウスの左クリックにより始点と終点を指定することで、指定された描画ソースとペンで直線を描画します。始点を選択したあとでも、右クリックすることで、始点をキャンセルすることができます。

3) 矩形/塗り潰し矩形

マウスの左クリックにより対角となる2頂点を指定することで、指定された描画ソースとペンで矩形あるいは塗り潰し矩形を描画します（塗り潰しはパターンペンは利きません）。1頂点を指定したあとでも、右クリックすることで、その頂点をキャンセルすることができます。矩形と塗り潰し矩形はアイコンをダブルクリックするとトグルで切り換わります。

4) 多角形/塗り潰し多角形

マウスの左クリックにより頂点を次々と指定し、最後の頂点で左ダブルクリックすることで、指定された描画ソースとペンで多角形あるいは塗り潰し多角形を描画します（塗り潰しでパターンペンは利きません）。右クリックすることで、直前に指定した頂点をキャンセルすることができます。多角形と塗り潰し多角形はアイコンをダブルクリックするとトグルで切り換わります。

5) 楕円/塗り潰し楕円

マウスの左クリックにより楕円の中心と半径を指定することで、指定された描画ソースとペンで楕円あるいは塗り潰し楕円を描画します（塗り潰しでパターンペンは利きません）。中心から水平方向の距離が水平方向半径となり、垂直方向も同様です。中心点を指定したあとでも、右クリックすることで、中心点をキャンセルすることができます。また、半径指定中にシフトキーを併用することで、画面上でほぼ真円になるような座標を指定することができます。楕円と塗り潰し楕円は、アイコンをダブルクリックするとトグルで切り換わります。

6) 塗り潰し

マウスをクリックすることにより、その点を内側とする閉じた領域を指定された描画ソースで塗りつぶします。アイコン上でダブルクリックすることで、ペイント設定ウィンドウが開き、各種設定が行えます。

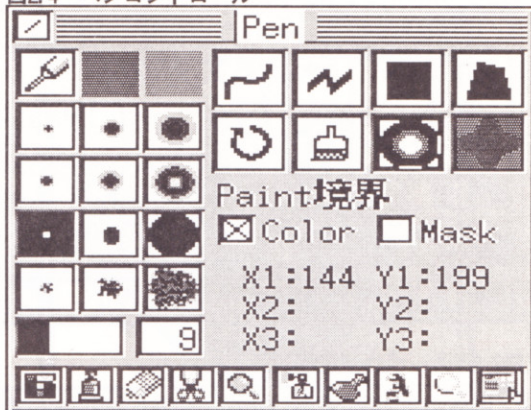
a) ペイント境界

境界を選択できます。カラーのみ、マスクのみ、カラー/マスク両方が選択できます。

b) 境界指定

特定カラーまたは特定マスクを境界としたい場合にチェックします。特定カラーまたは特定マスクの指定は、右のレベルバーで行い、ウィンドウ外で右クリックすることにより、作業画面上のカラー（ペイント境界がカラーのみ、およびカラー・マスク両方のとき）またはマスク（ペイント境界がマスクのみのとき）を設定することができます。

図2.4 ペンコントロール



c) 色変換

ペイント指定された点がカレントカラーになるように色が変換されます。境界指定をしない場合は、作業画面全体が対象となります。

なお、ペイント中に右クリックすることで、途中停止することができます。

7) グラデーション矩形/楕円

矩形または楕円と同じ要領で領域を指定したあと、グラデーションの中心となる点を指定することで、パレットコントロールで設定した中継グラデーションで矩形または楕円を描画します。中継グラデーションの左側が中心の色になります。描画ソースが裏画面カラー、裏画面マスクのときは動作しません。グラデーション矩形と楕円は、アイコンをダブルクリックするとトグルで切り換わります。

8) 平方/水平/垂直グラデーション

マウスの左クリックにより対角となる2頂点を指定することで、パレットコントロールで設定したグラデーション矩形を描画します（水平/垂直グラデーションは、中継グラデーションで水平方向および垂直方向に描画します）。描画ソースが裏画面カラー、裏画面マスクのときは動作しません。それぞれのグラデーションは、アイコンをダブルクリックすることで、順次切り換わります。

これらの描画機能は、グラデーションを除いて、CTRLキーを押しながらマウス操作を開始することで、アンドウパッファを描画ソースとしたアンドウペンとして機能します（アンドウパッファが確保されている場合）。

2. 5. 4. エディットコントロール

ハサミのアイコンをクリック、もしくはF4キーを押すことでオープンされます（図2.5）。ここでは編集方法を決定することができます。

・編集対象

編集時の対象を決定します。

・編集アイコン

編集方法を決定するアイコンです。

1. コピー

マウスの左クリックにより対角となる2頂点を指定することでコピー元矩形を指定し、さらに任意の点でクリックすることで矩形のコピーを行います。指定中でも右クリックをすることで、ひとつ前の指定をキャンセルすることができます。また、コピー元矩形を指定したあとにスペース

図2.5 エディットコントロール

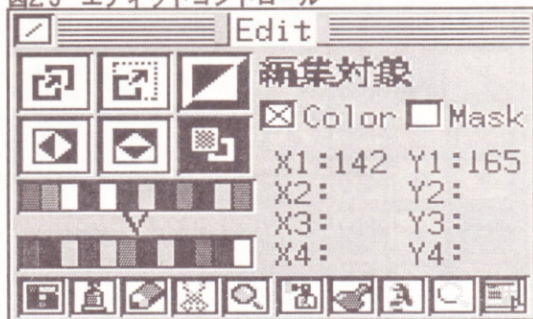
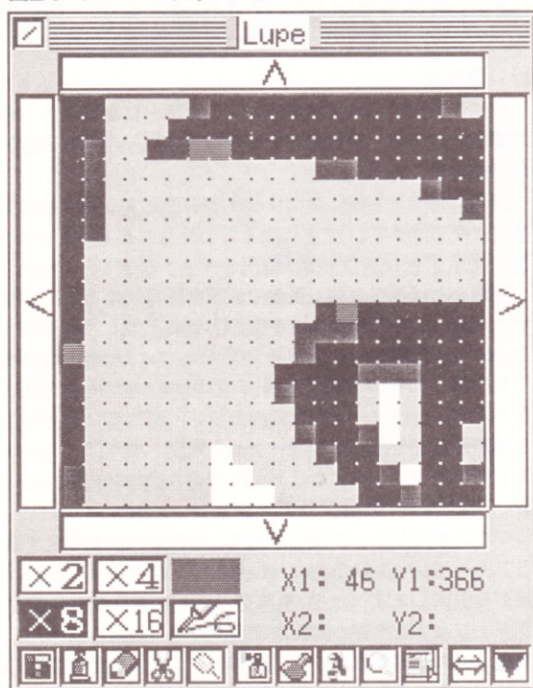


図2.6 ルーペコントロール



バーを押すことで表裏画面を反転させることができ、表から裏へのコピーを行うことも可能です。

2 リサイズコピー

マウスの左クリックにより対角となる2頂点を指定することでコピー元矩形を指定し、さらに転送先の矩形を指定することで矩形のリサイズコピーを行います。指定中でも右クリックをすることで、ひとつ前の指定をキャンセルすることができます。また、コピー元矩形を指定したあとにスペースバーを押すことで表裏画面を反転させることができ、表から裏へのリサイズコピーを行うことも可能です。

3 レベル反転

マウスの左クリックにより対角となる2頂点を指定することで、編集対象で指定された対象に対してレベル反転を行います。

4 左右反転

マウスの左クリックにより対角となる2頂点を指定することで、編集対象で指定された対象に対して左右反転を行います。

5 上下反転

マウスの左クリックにより対角となる2頂点を指定することで、編集対象で指定された対象に対して上下反転を行

表2.4 色変換の法則

| 対象 | C→C | C→M | M→C | M→M |
|-------------|------------------|---------------------|-------|-------|
| カラー | 画面のマスクを考慮して変換される | 動作しない | 動作する | 動作しない |
| マスク | 動作しない | マスクの下にカラーに対しては動作しない | 動作しない | 動作する |
| カラー ＆マスク | マスクを無視して変換される | マスクを無視して変換される | 動作する | 動作する |

います。

6 色変換

マウスの左クリックにより対角となる2頂点を指定することで、編集対象で指定された対象に対して色変換バーに従って色変換を行う。アイコン上でダブルクリックすることで色変換設定ウィンドウが開き、色変換バー編集を行うことができます。色変換バーはクリックすることで、カレントカラー/マスクでチェックされているほうをセットできますが、CLRがセットされているときはクリアされます。また、CLRボタンをダブルクリックすることで色変換バーを初期化することができます。カレントカラーからマスクへ、マスクからカラーへの変換も可能ですが、編集対象の選択方法によって色変換の挙動が表2.4のように変わります。重複するような変換が現れた場合は、色変換バーのより左にあるものを優先します。

2. 5. 5. ルーペコントロール

虫メガネのアイコンをクリック、もしくはF5キーを押すことでオープンされます(図2.6)。ここでは、作業画面の一部分をウィンドウ内に拡大して、拡大されたものに対して描画/編集することができます。

・ルーペ1/2

ウィンドウ外に表示された枠の中がここに拡大表示されます。この中でマウス操作を行うことで、描画/編集が可能です。

・スクロールボタン

拡大領域のスクロールを行います。また、ウィンドウ外に表示された枠を直接ドラッグすることでも拡大領域の移動ができます。

・ドットエディットボタン

クリックしてONになっているあいだはペンにはドットペンに、描画方法は自由曲線になります。もう一度クリックすることで解除されます。

・ダブルルーペ

ダブルルーペウィンドウが開き、同時に2つの独立したルーペを使用できます。

・ワイドルーペ

画面下にメニューバーが表示され、それ以外の画面全体がルーペになります。ルーペをスクロールさせるには、マウスカーソルを移動させたい方向の画面端に押しつけるように操作してください。

2. 5. 6. ブラシコントロール

霧吹きのようなアイコンをクリック、もしくはF6キーを押すことでオープンされます(図2.7)。

ここでは、エアブラシを吹き付けたときのようなリアルなイメージ画像を手軽に作成するためのシステムが提供されます。

・ブラシイメージ

ブラシのイメージ、濃度を確認できるウィンドウです。マウスでドラッグすることで、ブラシの大きさを変更できます。

・芯濃度

ブラシの中心の濃度を示します。

・描画目安

左が白地、右が黒地に描画した場合の中心の色の目安を示します。

・拡散形状

ブラシの拡散形状を3種から選択できます。

・フラクタルレベル

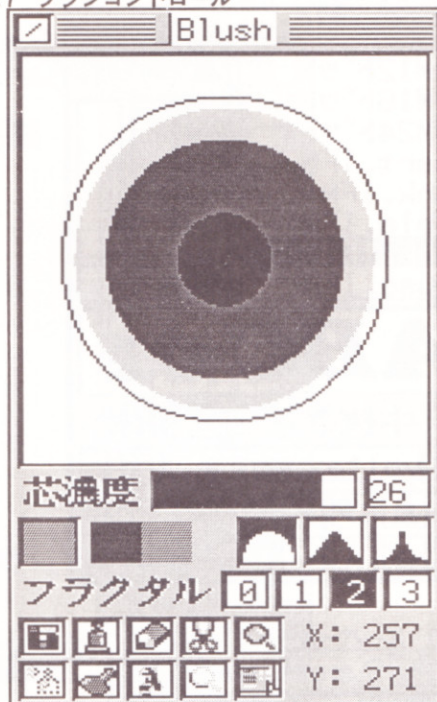
ブラシにランダムフラクタルを施すことができます。値が大きいほど強く、0はランダムフラクタルを施します。ブラシイメージボックスには反映されません。

・ウィンドウ外

左ドラッグすると、反転色で軌跡が表示され、ボタンを離すと軌跡通りにブラシでなぞられます（多少時間がかかります）が、ボタンを離すときにシフトキーを押しておくことで、描画を次に持ち越すことができます。描画はカレントカラーで行われますが、右クリックにより作業画面からカラーを拾うことができます。

なお、ブラシはブラシコントロール中でのみ使用できます。

図2.7 ブラシコントロール



2. 5. 7. スタンプコントロール

はんこのアイコンをクリック、もしくはF7キーを押すことでオープンされます（図2.8）。ここでは、スタンプ設定ディレクトリファイルEX_Stamp.SYSで定義されたディレクトリにあるスタンプパターンの描画、登録、削除を行うことができます。スタンプパターン登録時にアナログマスクを利用することで、半透明スタンプを作成することができます。

・スタンプパターン

すでに登録されているスタンプパターンを示します。クリックすることでカレントとすることができ、数が多いときは右のスクロールバーでスクロールして閲覧することができます。

・モードボタン

PutモードとGetモードを切り換えるボタンです。

・削除ボタン

ダブルクリックによってカレントのスタンプパターンを削除することができます。

・ウィンドウ外

1. Putモード時

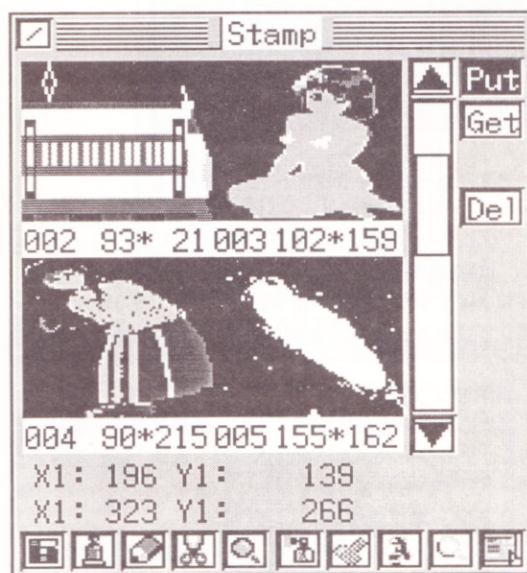
左クリックでカレントのスタンプパターンを作業画面に描画することができます。

2. Getモード時

左クリックにより、対角となる2頂点を順次指定することで、作業画面のその領域に描かれている絵をスタンプパターンに登録することができます。領域内にアナログマスクが施されている場合には、それに応じて半透明のパターンとなり、マスクレベル0の部分はスタンプ描画されません。

なお、スタンプはスタンプコントロール中でのみ使用できます。

図2.8 スタンプコントロール



2. 5. 8. フォントコントロール

文字のアイコンをクリック、もしくはF8キーを押すことでオープンされます（図2.9）。ここでは、ROMフォント、フォント設定ファイルEX_Font.SYSで定義されたSX-Window付属のIFMフォント、書体倶楽部のフォントを描画することができます。

・フォント名

使用できるフォント一覧です。クリックすることでカレントとすることができ、数が多いときは右のスクロールバ

図2.9 フォントコントロール

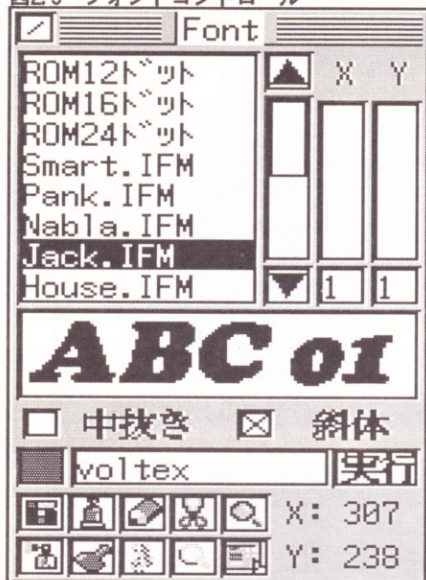
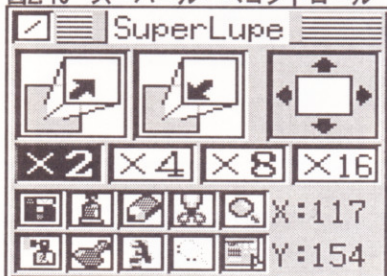


図2.10 スーパールーペコントロール



でスクロールして閲覧することができます。

・フォントサイズ

Xサイズ、Yサイズを独立して指定できます。1が16ドット、以降は1増えるごとに4ドットずつ大きくなりますが、IFMフォントは横幅が一定ではないので、Xサイズはこの限りではありません。また、ROMフォントはサイズ固定です。

・表示サンプル

書体の目安を表示します。また、書体倶楽部の場合は左から、アルファベット、平仮名、第一水準、第二水準をサポートしているかどうかを示しています。

・フォントオプション

ROMフォント以外は中抜き・斜体を指定できます。

・描画文字列

描画したい文字列を入力します。シングルクリックで編集、ダブルクリックで新規入力となります。

・実行ボタン

指定したフォント、サイズ、書体で文字列をカレントカラーでカーソルの位置から描画します。

・ウィンドウ外

左クリックすることで、ルーペカーソルをその位置へ移動させることができ、右クリックでカラーを拾うことができます。

なお、フォントはフォントコントロール中でのみ使用できます。

2. 5. 9. スーパールーペコントロール

黄金の虫メガネのアイコンをクリック、もしくはF9キーを押すことでオープンされます(図2.10)。ここでは、表示画面の一部分を拡大して全画面としたり、表示画面を裏画面の特定の位置に縮小してはめ込むことができます。裏画面が使用できない環境では、スーパールーペは使用できません。

・拡大

ウィンドウ外に表示されている枠を拡大して全画面とし、元の表示画面は裏画面に転送します。裏画面は破壊されます。

・復帰

表示画面を裏画面の枠が表示されている部分を拡大したものとして、縮小・はめ込みを行い表示画面とし、復帰前の画面を裏画面に転送します。

・スクロール

表示画面を裏画面の枠が表示されている部分を拡大したものとして、スクロールを行います。ワイドルーペ同様、マウスカーソルを画面端に押しつけることでスクロールできます。

・倍率

拡大・復帰時の倍率を指定します。

・ウィンドウ外

クリックすることで、拡大・復帰時の位置を示す枠を移動させることができます。

2. 5. 10. カスタムコントロール

コントロールウィンドウのアイコンをクリック、もしくはF10キーを押すことでオープンされます(図2.11)。主要なコントロールを集めたウィンドウです。

・描画ソース

左ダブルクリックすることで

カレントカラー → カレントマスク → リム・アマスク → 裏画面カラー

→ 裏画面マスク

と切り換わり(右ダブルクリックはその逆)、頻繁に使うもの2種を設定しておくことができます。

図2.11 カスタムコントロール

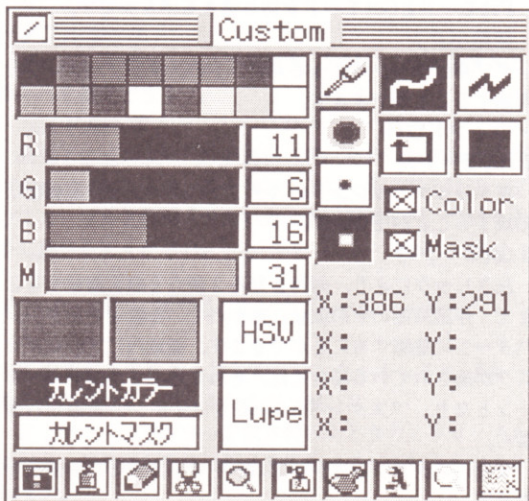
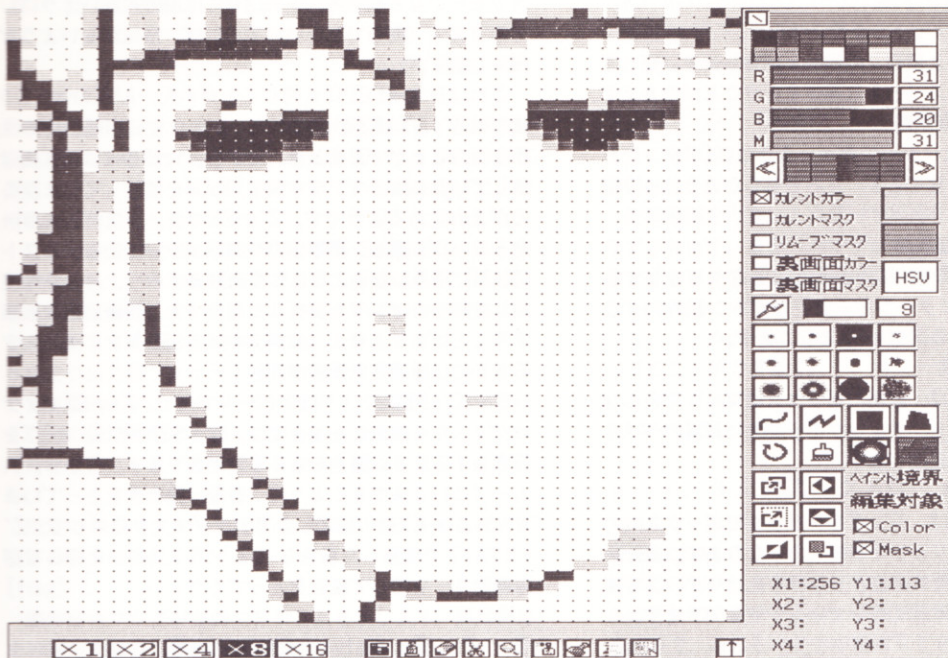


図2.12 カスタムルーペモード



・ペンアイコン

描画ソース同様、ダブルクリックすることで次々に切り換わります。頻繁に使うもの4種を設定しておくといでしょう。

・描画/編集アイコン

描画ソース同様、ダブルクリックすることで切り換わります。頻繁に使うツール4種を設定しておくといでしょう。

・ペイント境界/編集対象

カレントとなっている作業が塗りつぶしであればペイント境界を、編集であれば編集対象を示します。

・ルーペボタン

カスタムルーペを開きます(図2.12)。ルーペボタン上でマウスのボタンを押し、そのままドラッグすることで、拡大領域を示す枠がマウスについてきますので、任意の位置でボタンを放してください。操作法はワイドルーペと同様です。

2. 5. 1. 1. 特殊なコントロール

これまで解説してきたものは主に描画に関するコントロールだったわけですが、それ以外にもシステムの動作に関するコントロールがあり、以下のような起動キーに割り当てられています。

・HOMEキー

ダブルクリックのスピード、ウィンドウドラッグの方式の設定、オートフィックス設定と、EX-Systemの動作に関する各情報を得ることができます。

ダブルクリックのスピードは、クリック待ちのウエイト時間を表し、値が小さいほど、早くクリックしなければなりません。

・HELPキー

標準出力、および標準エラー出力になにか出力された場

図2.13 設定ウィンドウ (HOMEキー)

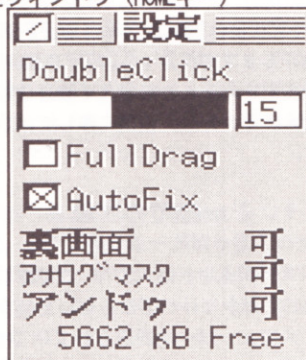
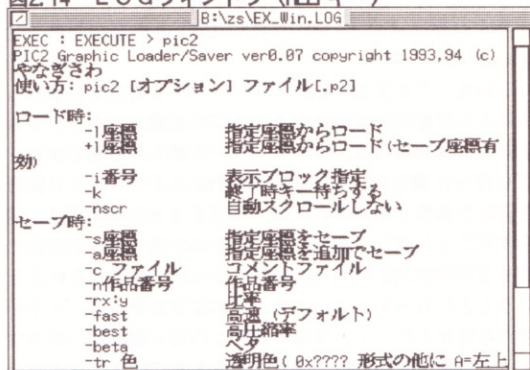


図2.14 LOGウィンドウ (HELPキー)



合、その内容を保存しておき、HELPキーを押すことで画面に表示します。

通常は参照する必要はありませんが、エラー発生時にはOSからのメッセージが出力されていることもありますので、ツールの動作がおかしいときなどには参考になるかもしれません。また、外部ファイルの開発中などには、デバッグの際に重宝することになるでしょう。

2. 6. 便利な操作法

ここでは、ちょっと便利な操作法を紹介します。知っておくと、スムーズでスマートな操作を行うことができます。

・確認ウィンドウ表示時

ESCキーもしくはリターンキー、またはOKボタンにカーソルキーを移動しなくとも、左右どちらかのクリックでOKとなります。

・選択ウィンドウ表示時

ESCキーまたはボタン以外のところで右クリックすることでもCancel、リターンキーまたはボタン以外のところで左クリックすることでもOKとなります。

・チェックボックス

チェックボックスは、厳密にボックスにマウスを合わせてクリックしなくても、チェックボックスに付属のラベルをクリックすれば、反応します。

・テキストボックス

テキストボックスは、シングルクリックでボックス内の文字編集、ダブルクリックで新規入力となります。また、文字入力時には、マウスの左クリックまたはリターンで確定、右クリックまたはESCキーでキャンセルできます。

2. 7. 注意

Z's-EXやMatier-EXは、Z's STAFFやMatierに寄生する形で動作しますが、Z's STAFFやMatierは、そういった形態で使用されることを考えて設計されているわけではありません。そのため、使用に際してはいくつかの注意が必要です。

2. 7. 1. Z's-EXでの注意

起動キー（デフォルトは‘S’）で起動するには、ウィンドウを閉じておかねばならないというのは先に述べましたが、メニューバーは表示されていてもZ's-EX側から消すことができます。メニューバーが表示されているかどうかは、キー割り込みを監視しておいて、スペースキーを押すたびにメニューバーのON/OFFを内部で判断させています。

しかし、この方法では、スペースキーを押さずにメニューバーの状態が切り換わったときには認識することができません。具体的には、メニューバーを消した状態でテロップを行った場合や、Z's STAFF ver. 3.0でコマンド実行を行った場合です。これらは、終了すると以前の状態に関らずメニューバーを表示してしまうので食い違いが発生し、Z's-EXを呼び出すと、必ずメニューバーが表示されるということになってしまいます。そのような場合は、もう一度同じ操作を行って、実際の状態と内部の認識を一致させてください（そういった操作をするときにはメニューバーを消さないように気をつけるのがいちばんですが）。

また、Z's STAFF ver. 1.0では、キー割り込みのタイミングの問題か、必ずしも正しく認識できないことがあるようです。

Z's STAFFは、待避画面（作業画面のバックアップのようなもの）として、テキストVRAMを使用しています。この待避画面が破壊されると、描画中の画像にとって致命

的なものとなりますので、システムエラーをトラップして、メッセージなどをテキストVRAMには出力しないようにしてあります（復帰できないのであれば、意味がないような気もしますが）。

これは日本語FEPにも同じことがいえ、ASKのver. 2.0まではZ's STAFFが自前のウィンドウで日本語FEPの操作を行うことができました。しかし、ver3.0には正式に対応していないようで、テキストVRAMを破壊してしまうことがまれにあるようです。Z's-EXでは、システムエラーをトラップし直し、標準出力や標準エラー出力もテキストVRAMに出力しないようにしてありますので、それらを使用しているコマンドを実行した場合でも、待避画面を破壊することはありません（直接テキストVRAMをアクセスするものには破壊されますが）。

Z's STAFFのマスクは1ビットでので、Z's-EXでアナログマスクを使用した場合は、Z's STAFFとZ's-EXを行き来するときに整合をとる必要があります。しかし、スムーズに行き来できるように、普段は整合性をとっていません。Z's-EX側で描画したアナログマスクに対して、Z's STAFFで編集した場合などは、OPT.1キーを押しながら起動キーを押して整合をとってください。

2. 7. 2. Matier-EXでの注意

Matierは、ver. 1.0と2.0でマスクの方式が違います。そのため、Matier-EXではMatierのバージョンを知る必要があるのですが、それを認識するのに、ベンダーファイルMAT.PENのサイズで判断しています。その判断を基に、Matier-EXを呼び出すときや復帰したときに整合をとっています。

Matierの裏画面のアドレスなどは、Matierを起動したあとに生成されるMAT. \$\$\$というファイルを参照することで得ています。このファイルや、先ほどのMAT.PENは、環境変数“MATIER”に設定されているディレクトリから探しますので、これらのファイルが見つからないというエラーメッセージが表示されたときは、“MATIER”に設定されたディレクトリに、それらのファイルがあるか確認してください。

なお、Matier側でも独自のアンドゥバッファを持っていますが、Matier-EXではそれを待避画面に使用しています。したがって、Matier-EXを呼び出した直後は、Matier側のアンドゥバッファは更新されている状態になります。また、Matier-EX側のアンドゥバッファも、オートフィックス設定に関らずMatierから移行した時点で更新されます。

3. 付属外部ファイルの機能

菊地 功

本書にはEX-System ver. 3.0専用の外部ファイルが多数収録されています。ここでは、その外部ファイルについて、付属コンフィグファイルに設定されている順に従って、使用法などの説明を行います。

ここで注意が必要なのですが、EX-Systemのメニューで表示されているのは機能名であり、多くは外部ファイルと1対1に対応しています。しかし、いくつかの機能は2つの外部ファイルの連続実行であったり、オプションによりひとつの外部ファイルが複数の機能を持ちあわせていることもあります。パラメータの設定などを変更したい方は、ここでの使用法をよく読んだあと、各自でコンフィグファイルをテキストエディタなどを使って編集し直してください。

指定できるパラメータなどは、外部ファイル名に続けて以下のように表記します。

[flag]

矩形指定フラグが有効であることを示します。矩形指定を省略した場合は、画面全体が対象となります。

[para: ?-?, ?]

パラメータが有効であり、パラメータの範囲と、省略時のデフォルトの値を示します。

[option]

オプションを指定できます。オプションの機能や、省略時のデフォルトは外部ファイルによって異なりますので、それぞれの項で説明します。

なお、EX-Systemはバージョンアップを重ねてきたシステムですので、各種ツールの作成にはいろいろな人の手関わっています。そういったものについては制作者名/原作者名を入れておきました。制作/原作者名が書かれていないものについては、私(菊地功)の制作によるものです。

また、再コンパイルにはGCCおよびXC ver. 2.0のライブラリが必要です。付属のメイクファイルはXC付属のMAKE. X用で、HAS. XおよびHLK. Xを使用しています。一部の外部ファイルでは、リンク時にXCのライブラリ以外に、本誌付属のEXLIB. Lおよび吉田泉氏のDESLIB. Oをリンクしなければなりません(ソースファイルおよびリンクの必要のあるものは、それぞれの外部ファイルの説明の最後に記述します)。

3. 1. ルート階層

いちばん最初の階層はシステムよりの機能とさまざまな階層から構成されています。以下にここで定義されている機能について説明します。

3. 1. 1. バージョン

●VerNo. x

現在稼動しているEX-Systemの名称と、バージョンを表示します。本誌に付属のEX-Systemであれば、すべてバージョンは3.0と表示されるはずですが。

[ポイント]

EX-System名はEXコール\$20のEXNUMから、バージョンは\$17のVERSIONから得ることができます。

[ソース]verno. s

3. 1. 2. 初期化

●Clear. x

作業画面をカレントカラーで初期化します。マスクはクリアされます。また、/wオプションをつけることで白で、/bオプションで黒で初期化することもできます。

[ポイント]

カレントカラーは、EXコール\$24のGETDRAWMODEから得ることができます。

[ソース]clear. c

[リンク]EXLIB. L

3. 1. 3. アスペクト比

●Aspect. x

768×512ドット表示モードに切り換えることで、ピクセルのアスペクト比を1:1で表示します。ただし、確認するのみで、このモードでの描画などを行うことはできません。Z's-STAFFはマウスカーソルをスプライトで描画していますが、ハードウェア上の制約で、768×512ドットモードではスプライトを表示できないからです。Matier-EXやEX-Windowでは、CLRキーを押すことでアスペクト比1:1のモードに移行し、そのまま描画などを行うことができます。詳しくは「2. 5. 使用法」を参照してください。

[ポイント]

表示モード切り換えには、CRTコントローラを直接操作しています。また、表示モード切り換えの前にEXコール\$22のMINTにより、マウスカーソルOFFや割り込みの停止を行っています。

[ソース]aspect. s

3. 1. 4. 実行

●Exec. x

任意の実行形式のファイルを呼び出すことができます。しかし、割り込みや画面周りの関係で破綻することもありますので、個人の責任において使用してください。外部ファイルもここから起動することは避けてください。

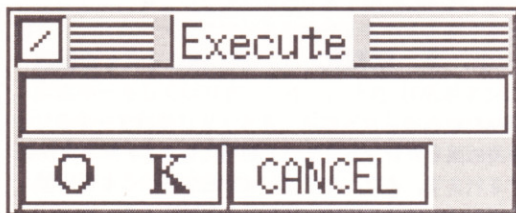
まず、図3.1のようなウィンドウが表示されますので、テキストボックスにコマンドラインを入力してください。正しく入力できたら、EXECボタンをクリックすることで、実行することができます。

[ポイント]

実行するコマンドとの割り込みの衝突を避けるために、実行前にEXコール\$22のMINTにより、マウスカーソルOFFや割り込みの停止を行っています。また、実行したコマンドにより画面モードが変更された場合は、to64k()関数によって65536色モードに変換されますが、実画面が1024×1024ドットの16色モードであった場合には、左上の512×512ドットが作業画面に変換されます。

[ソース]exec. c to64k. s

図3.1 外部ファイルの実行



3. 1. 5. 外部ファイラー

●ImgFile.x [option]

DOSの画像ローダ/セーバを呼び出すことで、EX-System上からさまざまな画像を扱うことができます。ただし、ローダ/セーバによっては割り込みや画面周りの関係で破綻することや、Z's-EXではマスクが破壊されることがありますので、新しくローダ/セーバを登録したときには、あらかじめ動作を確認しておくことをおすすめします。

まず、登録するローダ/セーバやオプションなどを設定するファイルを記述して、ImgFile.xと同じディレクトリに置く必要があります。設定ファイル名は、デフォルトではImgFile.SYSですが、オプションとして設定ファイル名を指定することもできます（拡張子は“SYS”固定でつける必要はありません）。画像フォーマットは8つまで設定することができ、ローダおよびセーバをそれぞれ定義することができます。ひとつの画像フォーマット設定は、以下の3行で構成されます。

```
[TITLE]
[EXT]
LOAD=[load command line]
SAVE=[save command line]
```

[TITLE]

画像フォーマット名を半角10文字以内で設定します。

[EXT]

半角3文字以内で拡張子を指定します。

[load command line]

ロード時の実行ファイル名およびオプションなどを記述します。実際のロードの際には、この後ろにファイラーで指定されたファイル名が付加されて、実行されます。ローダを設定しない場合は、NULを指定してください。

[save command line]

セーブ時の実行ファイル名およびオプションなどを記述します。実際のセーブの際には、この後ろにファイラーで指定されたファイル名が付加されて、実行されます。セーバを設定しない場合は、NULを指定してください。

設定ファイルには、このブロックを最大8つまで、続けて記述してください。本誌にもこの設定ファイルImgFile.r.SYSのサンプルが付属していますので、そちらも参考にしてください。

実行すると、図3.2のようなウィンドウが開きます。ファイラー部分の使用法は、はシステム側のファイルコントロールと同様です。

・画像フォーマットセレクト

扱いたい画像フォーマットのボタンをクリックすることで、そのモードに移行します。

・MASKチェック

チェックしておくことで、画像ロード時にマスクを有効にします。チェックしない場合は、マスクはクリアされます。

・オプション

ロード/セーブ時のオプションを示します。編集することも可能で、外部ファイラーを終了するまで有効です（設定ファイルには反映されません）。

・LOAD/SAVEボタン

指定ファイルを指定オプションでロード/セーブします。設定されていない場合は、文字が灰色で表示され、実行できないことを示します。

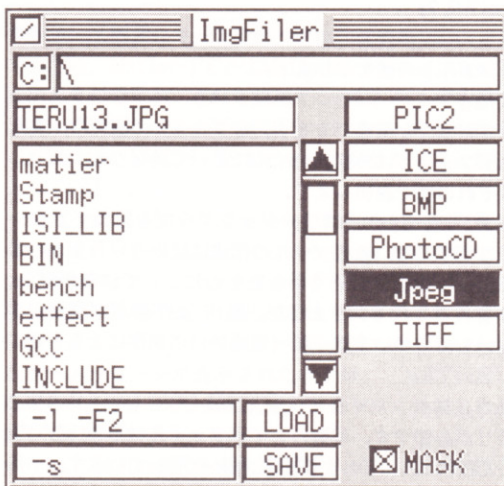
[ポイント]

Exec.x同様、ローダ/セーバを起動する前に割り込みを停止し、終了時にはto64k()関数で65536色モードに変換しています。これにより、256色データや、512×512ドットに限った16色のデータもロードすることができます。

[ソース]imgfiler.c to64k.s

[リンク]deslib.o EXLIB.L

図3.2 イメージファイラー



3. 1. 6. 表裏切り換え

●Alternate.x

(原作：丹明彦)

作業画面と裏画面を切り換えます。システム側のメニューでスペースキーを押したときと同じ動作をしますが、こちらはオートフィックスが有効になっていれば、作業画面のアンドウバッファへの転送も行われます。

[ポイント]

EXコール\$0AのALTERNATEモード0を使用しています。

[ソース]alternate.c

[リンク]EXLIB.L

3. 1. 7. 仮想画面

●VirScr.x

表示画面と512×512のサイズを超える仮想画面とのあいだのイメージ転送を行います。ここでの仮想画面はメモリ上に取られるのではなくディスクに保管されます。よって、Matierの仮想画面とは互換性はありません。しかし保管形式がGLM形式ですので、GLMフォーマットに対応したほかのアプリケーションから利用することができます。また、アナログマスク情報は、GMMというフォーマットで保管されます。

なんらかの操作を行うには、まずGLMファイルを指定する必要があります。すでに存在するGLMファイルを仮想画面として操作したい場合には、GLMボタンをクリックして、ファイラーからGLMファイルを指定し、LOADボタンをクリックしてください。

新規に作成する場合は、まずXSizeとYSizeを指定します。テキストボックスをクリックして数値を入力するか、ラベルを左右クリックして値を増減させ、作成したいサイズにしてください。その後、GLMボタンによりファイラーをオープンし、ファイル名指定後SAVEボタンをクリックしてください。これでディスク上に空の仮想画面が作成されます。

ファイルを指定すると、左上のビューポートに現在の仮想画面の内容が縮小表示され、それに対する作業画面が反転色のボックスで表示されます。このボックスはマウスでドラッグして移動させることができますし、その下のXPos、YPosの値（仮想画面の左上が原点）を変えることで移動できます。

右上のボタンは、仮想画面からビューポートに示された枠の部分を作業画面にロードし、その下のボタンは作業画面を枠の部分にセーブします。それに対し、その下の2つのボタンは、仮想画面全体と作業画面のあいだで、縮小ロード、あるいは拡大セーブを行います。このとき、どのように拡大/縮小が行われるかは、その下のモードボタンの状態で変わってきます。

・FULL

仮想画面全体が作業画面全体に対応します。

・1 : 1

仮想画面と作業画面のアスペクト比が同じであるとみなして、仮想画面の全体が作業画面に収まるような領域に対応します。

・4 : 3

仮想画面のアスペクト比が1 : 1であるとみなして（作業画面は4 : 3）、仮想画面の全体が作業画面に収まるような領域に対応します。

モードボタンは、クリックすることで順次変わりますので、実行前に設定しておいてください。

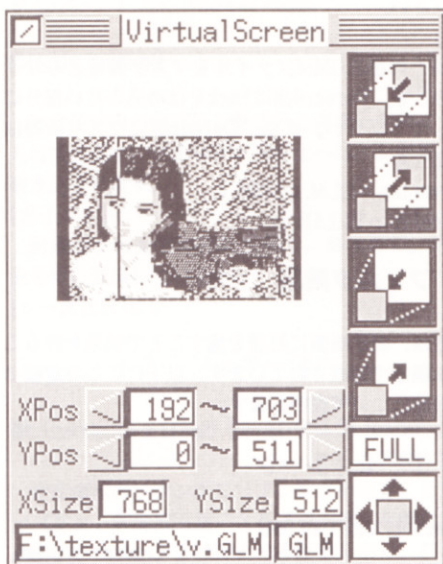
いちばん下は、仮想画面をスクロールして見るためのボタンで、メモリに十分な空きがある場合のみ使用できます。[ポイント]

アナログマスク情報を保管するGMMファイルは、マスク情報がある場合のみ生成され、GLMファイルと同じディレクトリに同じベース名で作成されます。また、読み込みに関しても、同じディレクトリに同じベース名のGMMファイルがあった場合に、そのGLMファイルのマスク情報として使用されます。

[ソース]virsr.c GLM.c GMM.c msmove.s

[リンク]deslib.o EXLIB.L

図3.3 仮想画面

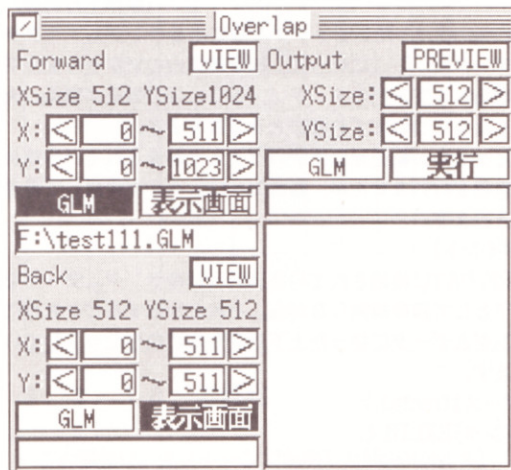


3. 1. 8. GLM合成

●Overlap.x

2つのGLMファイル（あるいは作業画面）を合成し、GLMファイルに書き出します。まずはForwardとBackを指定します。GLMファイルであれば、GLMボタンをクリックしてファイラーを呼び出し、そこからGLMファイルをロードしてください。作業画面を指定するのであれば、作業画面ボタンをクリックしてアクティブにしてください。この時点でGLMファイルを指定したのであれば、ForwardもしくはBackのラベルの右にあるVIEWボタンで、右下のビューポートに内容を縮小表示することができます。次にOutput側のGLMファイルのサイズを決定し、GLMボタンでオープンされるファイラーからファイルを指定します。ここでOutputラベル右のPREVIEWボタンをクリックすると、出力される内容が縮小表示されますので、それでよければ実行ボタンを、ForwardあるいはBackの位置を変更し

図3.4 GLM合成



なければ、それぞれの位置（Outputの左上が原点）を変更してください。

[ポイント]

仮想画面と同様にGMMファイルをマスク情報とみなします。このとき、Forward側はBackをはめ込みたい部分にマスクをしてください。また、Output側にマスク情報は出力されません。

[ソース]overlap.c GLM.c GMM.c

[リンク]deslib.o EXLIB.L

3. 2. フィルタ階層

この階層は、表示画面に処理を施すことで効果を得ることのできる機能で構成されています。以下にここで定義されている機能について説明します。

3. 2. 1. モノトーン

●Monotone.x [flag] [option]

(原作：丹明彦)

指定範囲をモノクロ32階調に変換します。

色の三原色RGBは、人間の目には緑がいちばん明るく、ついで赤、青の順に暗くなって見えますので、自然に見えるようにそれらの補正も行っています。

オプションはデフォルトでは/gのモノクロ変換ですが、/sを指定することで、セピア調に変換することができます。

アナログマスク上へは、レベルを考慮して書き込まれます。

[ポイント]

RGBの実際の比重は、青が28/256、赤が77/256、緑が151/256として処理しています。

[ソース]monotone.c g_monotone.s

3. 2. 2. セピア

Monotone.xのオプション/sを指定してたものです。

Monotone.xについては、前項を参照してください。

3. 2. 3. ランダムフラクタル

●Fractal.x [flag] [para:0-9,5]

(原作：丹明彦)

Fractal.xと同じディレクトリにあるランダムフラクタルデータRF.DATに従って、指定範囲内のピクセルを上下にずらしします。パラメータは、小さいほど効果が弱く、大きくなるにつれてずれ幅が大きくなります。

ピクセルの転送元にアナログマスクが施されている場合は処理を行いませんが、転送先へはレベルを考慮して書き込まれます。

[ポイント]

RF.DATに格納されている129×129のランダムデータを格子として線形補間しながら、それに対応するピクセルにランダムデータに従った上下ピクセルの色をセットしていきます。

[ソース]fractal.c

[リンク]EXLIB.L

3. 2. 4. 色相フラクタル

●FractalH.x [flag] [para:0-9,5]

同じディレクトリにあるランダムフラクタルデータRF.DATに従って、指定範囲内のピクセルの色相を変化させます。パラメータは、小さいほど効果が弱く、大きくなるにつれて変化が大きくなります。アナログマスク上へは、レベルを考慮して書き込まれます。

[ポイント]

Fractal.xと同様ですが、上下にずらす代わりにhsv()関数（標準ライブラリ）で得た色相をずらし、RGBtoHSV()関数で得た色をセットしています。

[ソース]fractalh.c

[リンク]EXLIB.L

3. 2. 5. フレア

●Flare.x [flag] [para1:0-9,6] [para2:0-9,5]

(原作：丹明彦)

指定領域内のマスク部分の下が光っているようなフレア処理を施します。アナログマスクはべたマスクと同様に扱われます。パラメータ1はフレアの強さ、パラメータ2は広がりを示します。アナログマスク上へは、レベルを考慮して書き込まれます。

[ポイント]

対象となるピクセルの近傍矩形内にあるマスクの面積から輝度を決定しますが、処理速度を稼ぐために、過去に調べたマスクの面積と今回との差を調べることで、面積を判断しています。

[ソース]flare.c

[リンク]EXLIB.L

3. 2. 6. フレア2

●Flare2.x [flag] [para1:0-9,6] [para2:0-9,5]

機能自体はフレアと同じですが、広がりを円弧で演算していますので、より自然なフレア処理を施すことができます。ただし、フレアに比べ若干処理時間を要します。

[ポイント]

あらかじめ作成しておいた円弧パターンに沿ってマスクの面積を調べるほかはFlare.xと同様です。

[ソース]flare2.c

[リンク]EXLIB.L

3. 2. 7. フレア3

●Twinkle.x [flag] [para1:0-9,6] [para2:0-9,5]

指定領域内のマウスで指定したピクセルの色を斜め方向に伸ばすことで、輝いているようなフレア処理を施します。マスクの扱いやパラメータはフレアと同様です。

[ポイント]

flare.xやflare2.xが対象ピクセルの近傍のターゲットを調べていたのに対して、Twinkle.xは矩形内のターゲットをサーチし、そこから斜め方向に色を拡散させています。

[ソース]twinkle.c

[リンク]EXLIB.L

3. 2. 8. 微分処理

この機能は、モノトーンで説明したMonotone.xと次に紹介するDiffer.xの連続実行により、構成されています。

●Differ.x [flag]

(原作：丹明彦)

青プレーンのX方向とY方向の輝度の差から、指定領域にレリーフのような処理を施します。青レベルの高い部分が飛び出して、左上から光を当てたような画像を得ることができます。アナログマスク上へは、レベルを考慮して書き込まれます。

[ポイント]

対象ピクセルの右下の青レベルからの差をとり、その値に応じた明るさのグレイをセットしています。

[ソース]differ.c

3. 2. 9. 裏表合成

●Compose.x [flag] [para:1-7, 4]

(原作：丹明彦)

指定領域の作業画面と裏画面を合成します。パラメータは裏画面の比率を表し、1/8単位で指定することができます。裏画面にアナログマスクが施されている場合には、そのピクセルに対する処理は行いませんが、作業画面はレベルを考慮して書き込まれます。また、裏画面の使用できないタイニーモデルでは動作しません。

[ポイント]

対象ピクセルのRGBそれぞれの作業画面のレベルをL0、裏画面をL1、パラメータをparaとしたとき、 $((L0 * (8 - para)) + (L1 * para)) / 8$ を新しいレベルとし、作業画面にセットします。また、比率が1:1 (パラメータ値4) のときは、RGBを展開せずに高速処理を行っています。

[ソース]compose.c

[リンク]EXLIB.L

3. 2. 10. 色強調

●Accent.x [flag] [para:0-6, 2]

(原作：丹明彦)

指定領域内の赤成分と緑成分と青成分のうち、もっとも値の大きい成分を強調し、コントラストの弱い画像をより原色に近づけます。パラメータの値を大きくするほど、処理後の画像は原色に近くなります。アナログマスク上へは、レベルを考慮して書き込まれます。

[ポイント]

変換前のピクセルの各要素をR0, G0, B0とすると、変換後の要素R1, G1, B1は次の式で与えられます。

$$\begin{pmatrix} R1 \\ G1 \\ B1 \end{pmatrix} = \begin{pmatrix} f1 & -f2 & -f2 \\ -f2 & f1 & -f2 \\ -f2 & -f2 & f1 \end{pmatrix} \begin{pmatrix} R0 \\ G0 \\ B0 \end{pmatrix}$$

$$f1 > 1, f2 = (f1 - 1) / 2$$

f1を大きくするほど原色に近くなります。

[ソース]accent.c

3. 2. 11. 色成分交換

●Cycle.x [flag]

(原作：丹明彦)

指定領域内の赤成分と緑成分と青成分をサイクリックに入れ換えます。アナログマスク上へは、レベルを考慮して書き込まれます。

[ポイント]

対象ピクセルをRGBに分解し、緑を赤に、青を緑に、赤を青に変換しています。

[ソース]cycle.c

3. 2. 12. ジャギー消去

●Linerevise.x [flag] [para1:0-9, 6] [para2:1-9, 5]

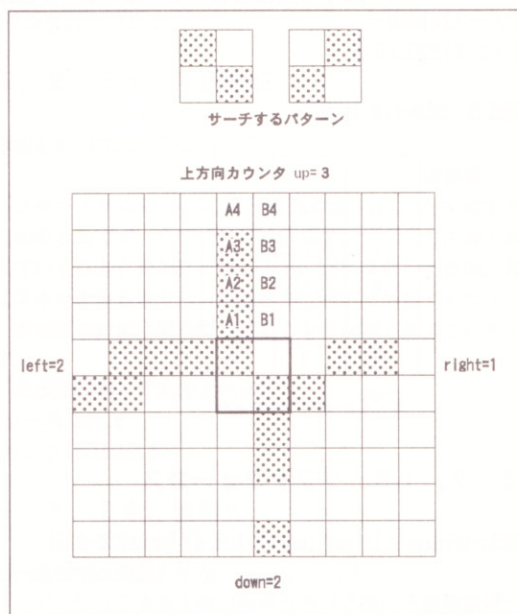
(原作：佐藤正春)

指定領域内にアンチエイリアシングを施すことで、ジャギー（ドット解像度不足によるギザギザ）を擬似的に消去します。パラメータ1は滲ませる距離を示し、パラメータ2は滲ませる濃度を示します。アナログマスクは、すべてべたマスクとして認識し、その下は処理を行いません。

[ポイント]

指定領域内を2×2マトリックスでサーチし、対角が同色のポイントを探します。仮に図3.5のようなポイントが見つかったとして、次にA1が同色かつB1が異色であった場合、A2が同色かつB2が異色であった場合、……とカウントしていきます。条件が満たされなくなった時点でカウントをやめ、カウント値と濃度からBxに滲ませる割合を決定し、下、右、左方向にも同様の処理を行っていきます。
[ソース]linerevise.c

図3.5 サーチパターンとカウント例



3. 2. 13. 二値化

この機能は、モノトーンで説明したMonotone.xと、次

に紹介するto2.xの連続実行により構成されています。

●to2.x [flag] [option]

指定範囲内の青プレーンの輝度から、モノクロ2階調に変換します。アナログマスク上へは、レベルを考慮して書き込まれます。また、/dオプションを指定することで、誤差拡散ディザをかけることができます。

[ポイント]

青レベルの閾値を16として、モノクロ2階調に変換しています。また、ディザオプションが指定されているときには、桑野式誤差拡散法により変換を行っています。

[ソース]to2.c

3. 2. 1 4. 二値化(Dither)

to2.xのオプション/dを指定しています。to2.xについては、前項を参照してください。

3. 2. 1 5. ガラス化

●Glass.x [flag] [para:1-9,4]

指定範囲内にぼかしをかけたあとに、ちょっとずらして原画との差を取ってやることで、ガラスのような質感を与えます。ロゴ作成などに有効です。パラメータはぼかしの強さを示し、大きくするほどのっぺりした感じになります。範囲指定の矩形は多少大きめに指定してください。また、矩形の左上を背景色とみなします。マスクは無視(破壊)されますので、注意してください。

[ポイント]


ぼかしは3×3のマトリックスで行っており、図3.6のような比重を持たせてあります。ぼかしをかける回数で強さを変えていますが、エッジの切り出しは1回ぼかしをかけた状態を参照しています。

[ソース]glass.c

[リンク]EXLIB.L

図3.6 ぼかしの比重

| | | |
|----------------|---------------|----------------|
| $\frac{1}{16}$ | $\frac{1}{8}$ | $\frac{1}{16}$ |
| $\frac{1}{8}$ | $\frac{1}{4}$ | $\frac{1}{8}$ |
| $\frac{1}{16}$ | $\frac{1}{8}$ | $\frac{1}{16}$ |

 ターゲットピクセル

3. 2. 1 6. ソフトフォーカス

●SoftFocus.x [flag] [para:1-9,4] [para2:0-9,5]

(原作：中野修一)

指定範囲内の明るい色を周囲に滲ませることで、ソフトフォーカス処理を施します。パラメータ1は滲ませる幅を示し、パラメータ2は明るい色と認識する閾値を示します。閾値は、小さいほど暗い色でも滲ませ、大きくすると、より明るい色しか滲まなくなります。アナログマスク下から

は滲ませる元となる明るい色を拾うことはしませんが、アナログマスク上への滲みは、レベルを考慮して書き込まれます。

[ポイント]

対象となるピクセルの近傍矩形内にあるピクセルで、一定以上の輝度を持ったものの和から輝度を決定しますが、処理速度を稼ぐために、過去に調べた輝度と今回との差を調べることで、総輝度を判断しています。

[ソース]softfocus.c

[リンク]EXLIB.L

3. 2. 1 7. クリスタルフレア

●Crystal.x [flag] [para:1-9,4] [para2:0-9,5] [option]

(原作：中野修一)

指定範囲内の明るい色をランダムに4方向に滲ませることで、輝くようなフォーカス処理を施します。パラメータ1は滲ませる幅を示し、パラメータ2は明るい色と認識する閾値を示します。また、オプション/hを指定することで、6方向に滲ませることもできます。アナログマスク下からは滲ませる元となる明るい色を拾うことはしませんが、アナログマスク上への滲みは、レベルを考慮して書き込まれます。

[ポイント]

斜めにスキャンすることで、SoftFocus.x同様に処理速度を稼いでいます。

[ソース]Crystal.c

[リンク]EXLIB.L

3. 2. 1 8. クリスタルフレア(Hex)

Crystal.xのオプション/hを指定しています。Crystal.xについては、前項を参照してください。

3. 2. 1 9. ぼかし

●EXShade.x [flag] [para:1-9,1] [option]

指定範囲内で、特定のピクセルに対して周囲の色の平均をとることで、ぼかし処理を施します。パラメータはぼかし範囲を示し、大きいほどぼかしが強くなります。

また、オプション/dを指定することで、2×2の単純ディザをかけることもできます。

アナログマスク上へは、レベルを考慮して書き込まれます。

[ポイント]

平均をとる領域は矩形ですが、2つの大きさの矩形の平均を使用することで、2段階の重み付けを行っています。また、処理速度を稼ぐために過去の履歴との差から濃度を調べています。

[ソース]shade.c

[リンク]EXLIB.L

3. 2. 2 0. ぼかし(Dither)

EXShade.xのオプション/dを指定しています。EXShade.xについては、前項を参照してください。

3. 2. 21. 色相変換

●HueMove. x

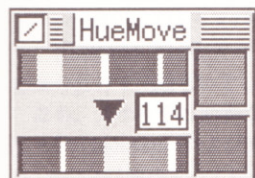
指定範囲内の色相を移動させることで、色の変換を行います。実行すると、図3.7のようなウィンドウが表示されます。上のグラデーションバーが変換前の色、下のグラデーションバーが変換後の色を示します。色相の移動量を指定するには、下のグラデーションバーをドラッグして左右にスクロールさせるか、テキストボックスにキーボードから移動量を直接入力します。右端の2つのボックスは、上が特定の変換前の色、下が上の色を変換したあとの色の目安を示し、上のグラデーションバーをクリック、もしくはウィンドウ外を右クリックすることで、その色を特定色としてセットすることができます。色相の移動量を決定したら、ウィンドウ外で左クリックにより矩形を指定することで、指定領域内の色相を指定移動量だけ変換します。アナログマスク上へは、レベルを考慮して書き込まれます。
[ポイント]

色相は0～191の範囲で表されますので、変換元の色相をh0、移動量をdh (≧0) とすると、変換後の色相hは、
$$h = (h0 + dh) \% 192$$

で表すことができます。このプログラムでは、まずRGBをHSVに変換し、Hを移動させたあと再びHSVからRGBに変換しています。ここで注意が必要なのは、RGBとHSVが完全に1対1に対応していないということです。そのため、変換の都度、誤差が生じてしまい、完全な可逆変換というわけにはいきません。

[ソース]HueMove. c RevBox. s
[リンク]deslib. o EXLIB. L

図3.7 色相変換



3. 2. 22. モーションブラー (接近)

●MOTION. X [para1:1-9,5]

(制作: 中野修一)

カメラと被写体の距離が変化しているときのぶれをシミュレートします。中心点を指定すると、その周りに動きのあるぶれを作り出します。

パラメータはぶれの強さを表します。大きなぶれはエッジがはっきりしすぎて綺麗に作れませんので、一度軽くぶれをかけてから行ってください。

アルゴリズムを簡単に解説します。基本は画像の合成です。極論すれば、処理は何点かの点の色の平均を取っているだけです。とある点から中心に向かって16個のサンプル点を取ります。これらの点の散らばり方は、描画を行う点からの距離に比例してまばらになり、その傾向は周辺部にいくほど大きくなります。逆に中心に近い部分ではサンプル点のほとんどはその点自身になります。

プログラムを変更して点の取り方を逆方向にすると被写体から遠ざかっているような効果も作れます。ちなみに、この場合は画面外の点をサンプルする際のクリッピングを行わなければなりません。

必ず全画面に対して実行されます。

[ソース]motion3. c
[リンク]EXLIB. L

3. 2. 23. モーションブラー (回転)

●vortex. x [para1:1-9,5] [para2:1-9,5]

(制作: 中野修一)

MOTION. Xに回転を加えたものです。カメラの軸ぶれに相当するものをシミュレートします。

最初のパラメータが回転の強さ、次のパラメータが偏角の大きさを指定します、

処理の簡略化のため、ぶれ自体は回転していません。サンプル点から一定の偏角でぶれ要素がサンプリングされていきます。

必ず全画面に対して実行されます。注意してください。

[ソース]motionp. c
[リンク]EXLIB. L

3. 2. 24. モーションブラー (平行)

●PMotion. x [para1:1-9,5]

(制作: 中野修一)

カメラが平行に移動したときのぶれをシミュレートします。背景に使用すると流し撮りのような画像になります。

起動すると画面中央に十字ラインが現れます。その中心に対する角度でぶれ方向を決定します。必ず全画面に対して実行されます。

[ソース]pmotion. c
[リンク]EXLIB. L

3. 2. 25. 主線合成

●mlt. x [para1:1-9,5]

(制作: 中野修一)

スキャナから取り込んだ下描きを使ってCGを作成するためのものです。色つけた画像にアナログ値の主線を加えていくことを目的にしています。表に着色した画像、裏にスキャナから取り込んだ画像を入れて実行してください。表画面の画像に裏画面の明るさを掛け算で加えていきます。

全画面に対して実行されます。

[ソース]mlt. c
[リンク]EXLIB. L

3. 2. 26. 放射光

●rad. x [para1:1-9,5] [para2:1-9,5]

(制作: 中野修一)

一部で要望が高かったフィルタを作成してみました。

私(中野修一)の作ったプログラムですから、かなり簡素なリストになっています。それでも描画時の場合分けが4通り、描画モード6通りに対応してありますので、多少

込み入った構成になってしまいました。結局、仕事をしているのはradiation()関数とrad()の一部だけです。このままではプログラムを見てもどの程度の方に理解していただけるか不安がありますので、一応解説をしておきます。

まず、光に形などありませんから、プログラムはかなり適当なものになっています。動作を見ておわかりのようにライン描画が基本になっています。光線の真ん中だけ見ればデキの悪いラインルーチンになっているはずですが、ただ、通常ドットを打つ処理をする部分がX軸方向への輝度減衰型水平ラインルーチンになっているだけです。光は一定の角度だけ拡散するという仕様で処理が行われており、角度に応じて減衰率が変わります。

本来なら、水平ラインではなく、光の角度から垂直になるラインを引いたほうが望ましいのですが、どうせ厳密な処理ではないので水平線で近似しています。また、角度を分類して縦横の処理を入れ換えるなどすれば画像のクオリティも上げることはできるはずですが、まだまだいくらかでも改造の余地はあります。このプログラムでは6つの動作モードに分かれていますが、どれも暫定的に決めたものですので、興味のある人は手を加えてみてください。

・使い方

通常の放射では光線の減衰などはありません。ちなみに、各項目とも第1パラメータは光の広がる角度です。

続いて、減衰系の指定です。光源から遠くなるほど光が弱まる「減衰」と、逆に中心に近づくほど暗くなる「集中」の2種類があります。それぞれ、減衰パラメータは減衰率の逆数だと思ってください。値が大きいほど長く光が伸びます。

「散乱」というのは、光源からの距離に応じサインカーブで輝度が変化するという仕様になっています。最初のパラメータは広がり大きさ、2つ目は変化の周期を表しています。周期を変えて重ねて使用することを前提としていますので、座標値はメモしておき、根気よく続けてください。相手は乱数ですのでときどきセーブするほうがよいと思います。

「彩色」と「極彩」は散乱の変化周期をRGBで位相を変えたものです。

それぞれ、起動時にシフトキーが押されていると光源指定を画面外に設定できるようになります。中央にある四角が画面ですから、それを参考に位置を指定してください。ただし、減衰系のもものでは、指定によっては光が画面内まで届かないこともありますので注意してください。

[ソース]rad3.c
[リンク]EXLIB.L

3. 2. 27. 質感合成

●Attribute.x

表画面にあるテクスチャの素材感を裏画面の画像と合成します。表裏を間違えやすいので注意してください。

表画面の画像はモノクロ化され、画面内の平均的な輝度を求め、全体の明度バランスを見て裏画面の画像と明度を合成していきます。

テクスチャの指定はPicPaintをブラウザとして使用するとよいでしょう。

[ソース]Attribute.c
[リンク]EXLIB.L

3. 2. 28. 明度調整

●Bright.x

画像のG成分(緑)を輝度とみなして、値を増減させます。

[ソース]Bright.c
[リンク]EXLIB.L

3. 2. 29. 明度加減算

●AddLevel.x

画像のG成分(緑)を輝度とみなして、値を増減させます。主にプレーンごとに分離したデータの加工などに用います。

[ソース]AddLevel.c
[リンク]EXLIB.L

3. 2. 30. TWIRL

●TW3.x

(制作: 中野修一)

指定範囲の中心に向かって渦巻変形を行います。

[ソース]TW3.c
[リンク]EXLIB.L

3. 2. 31. うのように

●TWIRL2.x

(制作: 中野修一)

TWIRLのバリエーションで、渦巻しながら奇怪な変形を行います。

[ソース]twirl2.c
[リンク]EXLIB.L

3. 2. 32. 収差変形

●Shusa.x

(制作: 中野修一)

画面を糸巻き型、樽型にカメラのレンズのように収差変形します。

[ソース]Shusa.c
[リンク]EXLIB.L

3. 2. 33. 横波

●SinWave.x

(制作: 中野修一)

画面をSIN関数で横に揺らします。

[ソース]SinWave.c
[リンク]EXLIB.L

3. 2. 34. 横波2

●SinWave2.x

(制作: 中野修一)

画面をSIN関数で画面中央を中心として横に拡大/縮小

して揺らします。

[ソース]SinWave2.c

[リンク]EXLIB.L

3. 3. ツール階層

この階層は、直接画面を操作する機能などで構成されています。以下にここで定義されている機能について説明します。

3. 3. 1. 裏画面パース変形

●Perspective. x

(制作：丹明彦)

PERSPECTIVE. Xは裏画面の内容を拡大縮小および回転させ、パースをつけて表画面に描画します。裏画面1枚分だけを描くモードと繰り返して描くモードを持っています。また、拡大縮小回転に伴う画質の荒れを抑えるために色を補間して描画するモードも備えています。

1ビットマスクおよび8ビットマスクの両方に対応しています。色補間モードにおいては、マスクの値も同時に補間します。

[インストール]

Zs_EX. SYSまたはEx_Win. SYSのコマンドラインに、

PERSPECTIVE. X [/F] [/B]

と書きます。スイッチの機能は次のとおり。

/F …… 全画面に繰り返し描画する

/B …… 色を補間して描画する

デフォルトは1枚描画、および補間なし描画。4通りの組み合わせが存在しますが、使いたい機能の数だけコマンドラインを記述する必要があります。なお、色を補間しても(スイッチ/B)極端に速度は低下しないので、実用上は画質の上がるモード。

PERSPECTIVE. X /B

PERSPECTIVE. X /F /B

の2つだけで問題ないでしょう。

なお、スイッチの文字の由来はそれぞれ、全画面を意味するFullscreen、および双線形補間を意味するBilinear interpolationによっています。

また、RESETを押して初期状態に戻したあとに実行すると、専用の2次元転送により、高速に裏画面から作業画面に転送することができます。2次元転送を除いて、裏画面のアナログマスクはすべてベータマスクとして認識し、作業画面はアナログマスクとしての透過率を考慮されます(2次元転送は裏画面のアナログマスクも考慮されます)。裏画面を使用できないタイニーモデルでは動作しません。

[使い方]

PERSPECTIVE. Xを起動すると、

- ・操作パネル
- ・フレーム(裏画面の場所を示す枠)が現れます。操作パネルには次のようなボタンがあります。
- ・回転/平行移動ボタン
フレームを動かし、裏画面が描画される領域を指定します。マウス右ボタンを用いると回転および平行移動の量が左ボタンを用いた場合の1/4になります。
- ・OKボタン
裏画面の内容を表画面に描画します。

・RESETボタン

フレームを初期位置に戻す。回転や移動を繰り返して混乱した場合に用いるとよいでしょう。

[生成ファイル]

PERSPECTIVE. EXというファイルに回転および移動の状態を保存します。

[ポイント]

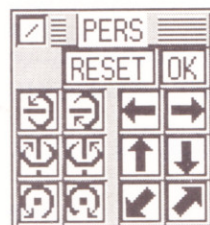
3次元の回転や平行移動のマトリクス演算を行いますが、パースをつけるために、得られたXやY座標をZ(奥行き)で割らなければなりません。ただし、実際問題としては、転送元のピクセルを転送先のピクセルに対応させていたのでは、過不足が生じてしまうので、その反対の逆変換を行っています。また、処理速度を稼ぐために、実数演算を控え、256倍の下駄を履かせた整数演算で代用しています。

[ヘッダ]typedef.h mat3.h

[ソース]perspective.c pers.c mat3.c RevLine.s

[リンク]EXLIB.L

図3.8 パースペクティブ変形



3. 3. 2. スクロール

●Scroll. x

表示画面全体をスクロールさせます。実行すると、図3.9のようなウィンドウが開きます。対象を個々に選択することで、カラーとマスクを別々にスクロールさせることもできます。スクロール値は、テキストボックスをクリックして、キーボードから値を入力することもできますし、ウィンドウ外でマウスをドラッグすることでも、指定することもできます。スクロール値は、スクロールを終了するまで保持されます。

[ポイント]

カラーとマスクを別々にバッファ内でスクロールしておいて、最後にEXコール\$2DのBUF2GRAMで表示します。

[ソース]scroll.c roll.s

[リンク]deslib.o EXLIB.L

図3.9 スクロール



3. 3. 3. パレット

●Palet.x

システム側のパレットの編集を行うことができます。これ自身は作業画面の編集などを行うことはできませんが、次項で説明するCGrad.xから呼び出すことができます。実行すると、図3.10のようなウィンドウが開きます。上の16のパレットはシステムと共通で、ここでのパレットの更新はシステムに反映されます。カレントパレットは反転色のボックスで示され、その要素が下のレベルバーに示されます。また、レベルバーを操作してカラーを作成したり、ウィンドウ外で右クリックすることで、その座標のカラーをカレントパレットにセットすることができます。右下のボタンもクリックすることでスポイトモードとなり、カラーを拾うことができます。

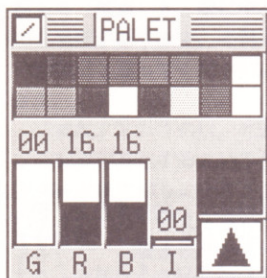
[ポイント]

システムパレットの設定、取得、アクティブパレットの設定は、EXコール\$18のSETCOL、\$19のGETCOL、\$1AのACTPALで行うことができます。

[ソース]palet.c

[リンク]EXLIB.L

図3.10 パレット



3. 3. 4. 曲グラデーション

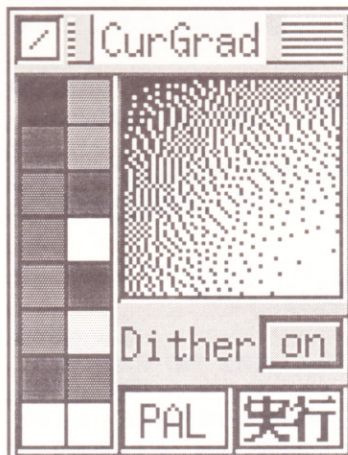
●CGrad.x [flag]

指定領域内に境界が円弧を描くグラデーションを描画します。実行すると、図3.11のようなウィンドウと、円と四角い枠が表示されます。ウィンドウ内の16のパレットはシステムと共通で、左クリックでカレントパレットの選択、右クリックでカレントパレットへの色のセットです。色が決定したら、右のモノクロディザで描画されたグラデーションにセットします。左上端のほうに円弧の外側、右下端に円弧の内側の色を、左クリックでセットしてください。また、パレットの色を任意に編集したい場合は、PALボタンで前項で説明したPalet.xを起動することができます。カーブの度合や方向を決定するには、ウィンドウ外で設定したい円弧の中心を左クリックして決定してください。円はカーブの目安となります。また、ディザが必要ない場合は、Ditherボタンをクリックして、offの状態にしておきます。アナログマスク上への描画は、透過率を考慮されて作業画面に反映されます。

[ポイント]

ディザは桑野式誤差拡散法で行っています。また、処理速度を稼ぐために、実数演算を行わずに、下駄を履かせた

図3.11 曲グラデーション



整数演算で代用しています。

[ソース]cgrad.c RevBox.s RevEllipse.s

[リンク]EXLIB.L

3. 3. 5. 拡大・縮小

●Zoom10.x

特定領域を全画面に拡大、または全画面を特定領域に縮小します。制約がありますが、システムの拡大・縮小よりも高速です。実行すると、図3.12のようなウィンドウが開きます。拡大または縮小を選択したあとに、ウィンドウ外で領域を指定してください。シフトキーとの併用で、アスペクト比を一定にすることができます。転送元マスクは無視されますが、転送先マスクは透過率を考慮されて描画されます。

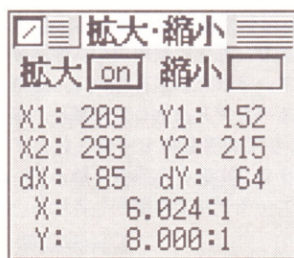
[ポイント]

サイズに合わせて適当な下駄を決定することで、整数演算のみで正確、かつ比較的高速に変換することができます。

[ソース]zoom10.c RevBox.s

[リンク]EXLIB.L

図3.12 拡大/縮小



3. 3. 6. プレーン

●Plane.x

裏画面の特定のプレーンを作業画面に抽出、または裏画面の特定のプレーンを作業画面に合成します。実行すると、図3.13のようなウィンドウが開きます。操作できるプレーンは、赤 (R)・緑 (G)・青 (B)・色相 (H)・彩度 (S)・明度 (V) の6種類です。抽出時はR・G・B・

S・Vはグレースケールで作業画面に描画されますが、HはSとVが100%とした場合の色で描画されます。合成時は、R・G・B・Hはそれぞれの値を作業画面のそれと入れ換えますが、SとVは青の輝度をそれぞれの値とみなして合成します。裏画面のマスクは無視されますが、作業画面のマスクは透過率を考慮されて描画されます。裏画面を使用できないタイニーモデルでは動作しません。

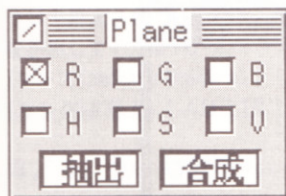
[ポイント]

裏画面を直接操作してしまうと、アンドウが利かなくなってしまうので、抽出・合成の向きは常に裏→表になっています。

[ソース]plane.c

[リンク]deslib.o EXLIB.L

図3.13 プレーン分離/合成



3. 3. 7. PICペイント

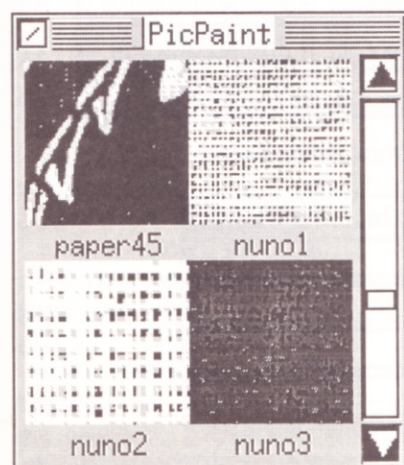
●PicPaint.x

あらかじめ登録されているPICフォーマットの画像で、カラー境界の領域をペイントします。画像を登録するには、まずPicCut.x（外部ファイルではありません）を使って、インデックス画像を作成します。コマンドラインから、

PicCut filename[.pic]

と入力して実行すると、指定されたPICファイルの左上64×64ドットが切り出され、filename.IDXというファイルが作成されます。このインデックスファイルとPICファイルを同じディレクトリに置き（ベース名が同じであれば、インデックスとPICファイルの内容が同じである必要はありません）、PicPaint.xと同じディレクトリにあるPicPaint登録ファイルPicPaint.SYSにテキストエディタなどで、フルパス（拡張子は必要ありません）で記述します。

図3.14 PICペイント



複数のPICを登録するときには、それぞれをリターンコードで区切って記述してください。これで、PICペイントから、いま登録したPIC画像を使用できるようになりました。

PicPaint.xを実行すると、図3.14のようなウィンドウが開きます。まずは塗り潰したい領域を、左クリックしてペイントします。そのあと、ウィンドウ内のインデックスからペイントしたい画像を選びクリックすることで、選択した領域をPICでペイントすることができます。また、PICペイントを終了するまで、再びインデックスをクリックすることで、領域の塗り直しをすることができますし、新たに領域を追加することもできます。アナログマスクは透過率を考慮されて描画されますが、作業画面に半透明マスクがある場合に限り、処理の都合でPICペイント終了時に、もう一度ペイントが行われます。また、PICペイントでは、スペースキーによる表裏反転や、XF1キーによるマスク透視などを行うことはできません。

[ポイント]

ペイントする領域を保存するのに、1ピクセル1ビットのバッファを使用しています。PIC画像をロードしたあと、このバッファに基づいて退避画面から画像を復帰させています。

[ソース]PicPaint.c xpic.s

[リンク]deslib.o EXLIB.L

3. 3. 8. 画面回転

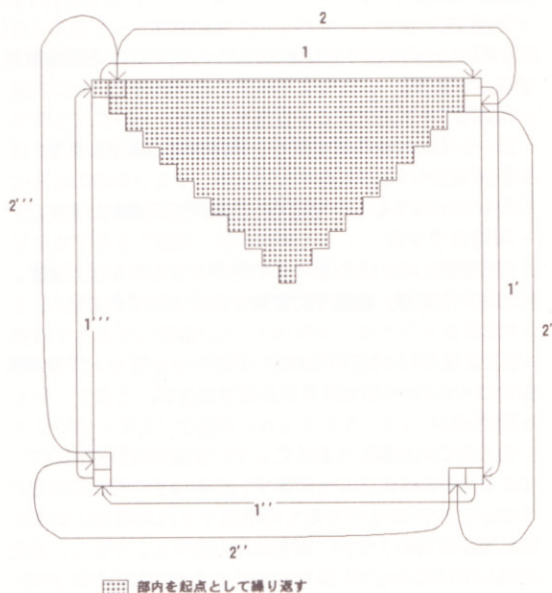
●Rot.x

作業画面を時計回りに90度回転します。マスクも同時に回転します。

[ポイント]

図3.15のように、ある転送元ピクセルを回転させた先のピクセルを再び転送元とし、順次回転させることで、余計なバッファを使用することなく画面全体を回転させることができます。この作業を作業画面、待避画面、マスクバッ

図3.15 回転ピクセルの転送



ファに行っています。
[ソース]rot.s

3. 3. 9. AMI

●AmiEX.x

福嶋章太氏によるアニメーションマルチイメージAMIの簡易編集ツールです。AMIについては、そちらの説明を参照してください。

実行すると、図3.16のようなウィンドウが開きます。

・ファイルボックス

現在編集しているAMIファイルを示します。クリックすると、図3.17のようなファイラーが開きます。ファイル名を指定したあと、すでにあるファイルを再編集したのであればLOADボタンを、新規作成する（あるいは既存のファイルを最初から編集し直したい）のであればSAVEボタンをクリックしてください。ただし、SAVEの場合は、あらかじめ同期カウンタ、再生モードを設定しておいてください。

・カレントコマナンバー

現在何コマ目を編集しているかを示します。左右のボタンで、カレントコマを移動することができます。

・再生モード

現在編集している再生モードを示します。編集に変更することはできません。クリックすると、図3.18のようなウィンドウが開きます。この中から、再生モードを選択でき、そのあとで新規作成した場合に有効となります。

・表示ボタン

カレントコマを作業画面に表示します。

・上書ボタン

現在の作業画面をカレントコマに上書きします。ただし、作業画面が再生モードで設定されている色数以下に減色されていなければなりません。

・追加ボタン

AMIファイルにフレームを追加し、現在の画面を追加したフレームの最初のコマに書き込みます。色数の制限は、上書きと同様です。

・>>ボタン

AMIファイルを再生します。Z's-EXでは、作業画面が破壊されます。

・《ボタン

カレントコマを1コマ戻し、作業画面に表示します。

・》ボタン

カレントコマを1コマ進め、作業画面に表示します。

・同期カウンタ

現在編集しているファイルの同期カウンタを示します。再生モード同様、編集に変更することはできません。

・PICボタン

PICFILERを呼び出します。PICフォーマットで表示画面のロード/セーブを行うことができます。

・EXボタン

あらかじめ登録しておくことで、任意の実行形式のファイル呼び出すことができます。これは主に減色処理を行うファイルの実行を考えての機能です。AMIは、記録の際に縮小は行いますが、減色は行いません。そこで、減色に関してはほかのツールでやってもらうことになります。固定パレットモードについては、2、4、16、256色のパ

図3.16 AMI

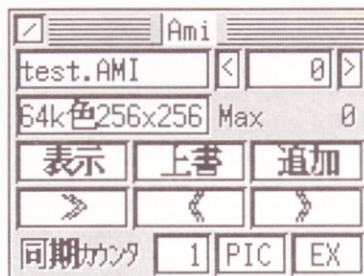


図3.17 AMIファイラー

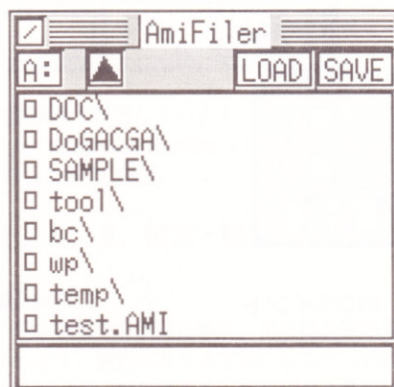


図3.18 AMIのモード選択

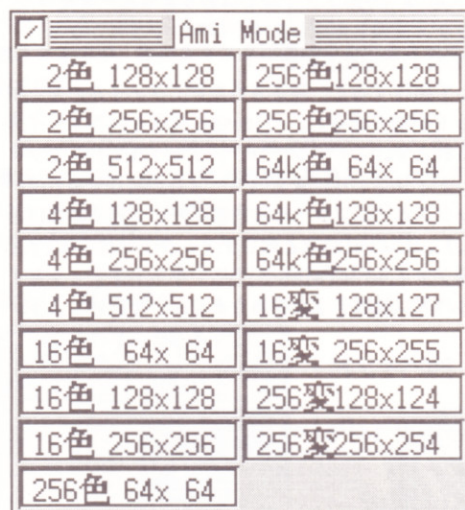
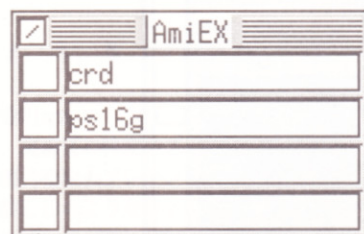


図3.19 外部ファイルの指定



レットとしてそれぞれAmiEX.xと同じディレクトリにあ

るMONO.PAL, G2T4.PAL, TONE16.PAL, G3R3B2.PALを参照するようになっています。

クリックすると、図3.19のようなウィンドウが開きます。右側の細長いボックスがコマンドボックスで、最大4つまでのコマンドを設定できます。コマンドボックスをクリックしてキーボードからコマンドを入力したら、その左のボックスをクリックすることで、実行することができます。ただし、コマンドによっては割り込みや画面周りの関係で破綻することや、Z's-EXではマスクが破壊されることがありますので、新しく登録したときには、あらかじめ動作を確認しておくことをおすすめします。ここで登録されたコマンドは、AmiEX.xと同じディレクトリにAMIEX.DEFというファイルに保存されます。

[ポイント]

AMIファイルの操作に関しては、すべてAMIのライブラリで用意されている関数を使用していますので、そちらを参照してください。

[ソース]amiex.c to64k.s

[リンク]EXLIB.L AMILIB.L

3. 3. 10. 陰影精円

●ShadeEllipse.x

表示画面に対して、そこに球を置いたような陰影付けを行います(変形は行いません)。

実行すると、図3.20のようなウィンドウが表示されます。

・直接光強度

光源から直接オブジェクトに投射する光線の強度を示します。

・環境光強度

周囲から均一に投射される光の強度を示します。

・ハイライト強度

投射された直接光が、オブジェクトで反射される光の強さを示します。値を大きくすると、光沢のある質感を与えることができます。

・ハイライト収束率

ハイライトの大きさを示します。値を大きくすると、ハイライトは小さくなり、滑らかな質感を与えることができます。

・平行光ベクトル

投射する光線のベクトルを示します。画面の右方向がX、下方向がY、奥方向がZに対応しています。また、球がサンプル表示されているボックス内でクリックすることでも、ベクトルを設定することができ、右クリックで逆光になります。

・影反転

チェックすると、影の部分が明るくなります。ハイライトは変化しません。全体を光の球のような効果に仕上げることができます。

各パラメータの設定を行ったあとに、ウィンドウ外で左クリックにより楕円領域を指定することで、その領域に陰影処理を施します。

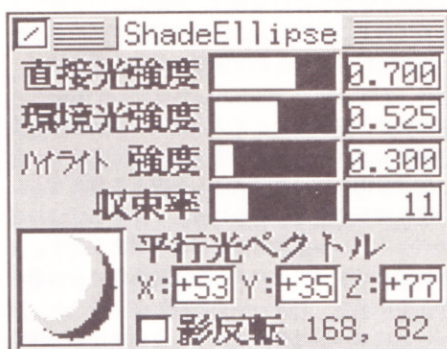
[ポイント]

真面目に演算を行っていますので、大きな領域に対して処理を施すには、かなりのマシンパワーと時間を要します。あとで説明する、3次元階層内の球状変形と処理内容はほとんど同じですので、そちらも参照してください。

[ソース]ShadeEllipse.c RevEllipse.s

[リンク]deslib.o EXLIB.L

図3.20 陰影精円



3. 3. 11. パターンブラシ

●PatternBlush.x

あらかじめパターンを登録しておくことで、そのパターンを表示画面にランダムに吹き付けることができます。まずはパターンを保存しておくディレクトリを、テキストエディタなどで記述し、PatternBlush.xと同じディレクトリにPatternBlush.SYSのファイル名でセーブしておきます。このファイルを作成しない場合は、PatternBlush.xと同じディレクトリにパターンの保存/参照が行われます。実行すると、図3.21のようなウィンドウが表示されます。ウィンドウ内には最大9つのパターングループが表示されており、それを超える場合には、左端のスクロールバーでビューポートをスクロールさせることができます。任意のパターングループをクリックすると、サイズ表示部分が反転し、指定したグループがカレントになったことを示します。

カラー選択ボタンは、左クリックするたびにDef→Cur→Alt→Defの順にモードが変わり(右クリックは逆順)、それぞれパターンが登録された時のカラー、システムのカレントカラー、裏画面カラーでパターンが描画されます。

軌跡ボタンはマウスの軌跡通りに、矩形ボタンは指定矩形内に対して描画を行うモードを指定し、Putボタンがアクティブになっているときにウィンドウ外でマウス操作を行うことで、表示画面にカレントグループのパターンをランダムに描画します。吹き付け半径は軌跡の場合のみ有効で、矩形で処理を行った場合には、マウスの右ボタンをクリックするまで描画し続けます。

登録できるパターンの大きさは最大64×64ドットで、ひとつのグループ内ではすべて同じ大きさで、最大32個まで登録できます。登録には、Getボタンをクリックしてアクティブにし、ウィンドウ外で最初のパターンを矩形で囲みます。すると、いま指定したパターンがウィンドウ内に新たなグループとして登録されますので、そのグループをダブルクリックしてください。すると、図3.22のようなウィンドウが表示され、現在そのグループに登録されているパターンが表示されます。ウィンドウ外にマウスを移動させると、パターンのサイズの枠がマウスと一緒に移動しますので、さらにパターンを登録する場合には、その枠に登録したいパターンに合わせてクリックしてください。正常に

登録された場合には、ウィンドウ内に反映されます。また、削除したいパターンにマウスカーソルを合わせてダブルクリックすることで、任意のパターンを削除することができます。

グループ自体を削除したい場合は、削除したいグループをクリックしてカレントにし、Delボタンをダブルクリックします。確認は行いませんので気をつけてください。
[ポイント]

システムのスタンプ同様に、アナログマスクを利用することで、半透明のパターンを登録することができます。ただし、ウィンドウ内にはべたマスクしか表示されません。

メモリの空き容量が少ない場合には、1グループにつき32個未満のパターンしか登録できない場合があります。また、登録されているすべてのパターンを読み込めない場合もあります。その場合、パターンの削除などを行うと、読み込めなかったパターンは破棄されてしまいますので注意してください。

[ソース]PatternBrush.c BrushGroup.c RevBox.s

[リンク]deslib.o EXLIB.L

図3.21 パターンブラシ

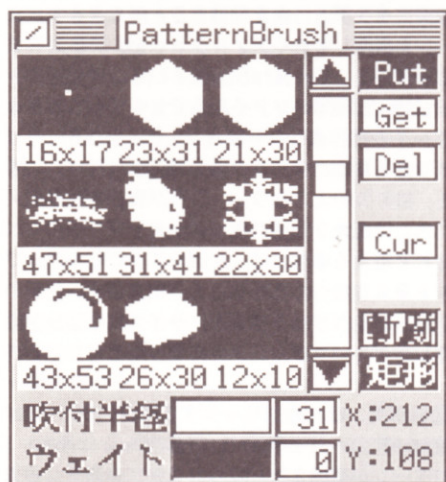
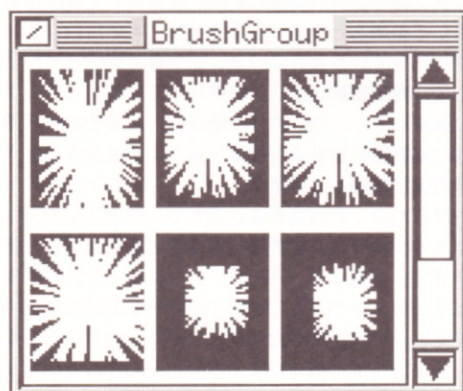


図3.22 パターンの登録



3. 3. 1 2. 回転

●Rotate.x

画面回転が画面全体の90度単位の回転であるのに対して、

こちらは任意の円内の任意角度の回転を行うことができます。最初に処理を行う円の指定を促してきますので、右下の座標を目安に円の中心、半径の順にマウスの左クリックで指定します。この円は、画面上ではほぼ正円になるように、X Yの半径比が3:4になります。また、指定中に右クリックすることで、キャンセルすることができます。円を指定すると、その円の中心から上に角度を示す直線が表示され、図3.23のようなウィンドウが開きます。ここで、回転角テキストボックスをクリックしてキーボードから角度を入力するか、ウィンドウ外で中心からの角度をマウスで指定してください。適当な角度が設定できたら、実行ボタンをクリックすることで、回転処理を開始します。元画像はRotate.x自身がバッファに持っていますので、何度でもやり直しが利きますし、取消ボタンで元に戻すこともできます。画面回転同様、マスクも回転が行われます。

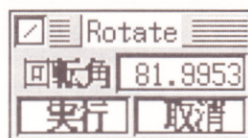
[ポイント]

適度なオーバーサンプリングを行っていますので、極端に線が途切れたり、がたがたになってしまうことはありません。しかし、それによって多少ぼけたような画像になってしまう場合があります。

[ソース]Rotate.c RevLine.s RevEllipse.s

[リンク]deslib.o EXLIB.L

図3.23 回転



3. 3. 1 3. 回転2

●Rotate2.x

円の外周は回転させず、中心にいくほど回転角が大きくなる以外は、Rotate.xと同じです。中心での回転角は、右周り360度未満のとなります。

[ソース]Rotate2.c RevLine.s RevEllipse.s

[リンク]deslib.o EXLIB.L

3. 3. 1 4. アンチエイリアスペイント

●aapaint.x [para:0-9,0]

境界がはっきりしない領域に対して、カレントカラーでペイントします。スキャナで取り込んだ線画などをペイントする際に有効です。パラメータは貫通性を表し、値が大きいほどノイズに強く狭い箇所も通り抜けることができますが、曖昧な境界は通り抜けてしまいます。通常は0~2程度が妥当でしょう。また、アナログマスクの下は処理を行いません。

[ポイント]

RGBそれぞれのレベルについて、いちばん下または上まで到達した時点を境界と認識し、そこまでを色相と明度を適当に変化させてペイントしていきます。パラメータは誤差であり、たとえば誤差1であれば、いちばん下まで到達しても1ずつならば登っていける、といったようなイメージです。

[ソース]aapaint.c
[リンク]EXLIB.L

3. 4. マスク階層

この階層は、マスク関連の機能などで構成されています。
以下にここで定義されている機能について説明します。

3. 4. 1. マスク操作

●Mask. x

作業画面全体に対して、マスクの操作を行います。システム側の操作により同じ結果を得ることもできますが、より手軽に操作することができます。実行すると、図3.24のようなウィンドウが開きます。

・VIEW

マスクを透視することができます。メニューでXF1キーを押すのと同等な働きをします。

・REV

マスクを反転します。アナログマスクに対しては、レベルが31から減じた値になります。

・ALTERNATE

表と裏のマスクを入れ換えます。裏画面を使用できないタイニーモデルでは動作しません。

・OFF

すべてのマスクをクリアします。

[ポイント]

表/裏画面のマスクバッファのアドレスは、それぞれEXコール\$1DのMASKADRのページ0/1で得ることができます。

[ソース]mask.c mask_.s
[リンク]EXLIB.L

図3.24 マスク操作



3. 4. 2. 緑→マスク

●GtoMask. x

作業画面の緑プレーンを、アナログマスクに変換します。緑要素が大きい部分ほど、マスクが強くなり（レベルは低くなる）、緑が31のところはべたマスク、0のところはマスクされません。変換により、以前の作業画面はマスクも含めて破壊されます。アナログマスクを使用できないスモールモデル以下では、マスク部分はすべてべたマスクに変換されます。

[ポイント]

緑レベルと31のXORをとって、マスクのレベルとして

います。

[ソース]gtomask.s

3. 4. 3. マジックワンド

●Wand. x [para1:0-9, 4] [para2:0-9, 4]

類似した色の周囲を、マスク（アナログマスクが使用できる環境ではマスクレベル16）で囲みます。パラメータ1は色相の閾値、パラメータ2は飽和度・明度の閾値を示します。色相の閾値は大きいほど広い色相を同色とみなしますが、飽和度・明度の閾値はグレイなどの色相で表せない色の範囲を示し、大きいほどそれに近い色をグレイとみなします。

実行すると図3.25のようなウィンドウが開き、画面左上に四角い枠が表示されます。ウィンドウの内部にはこの枠内が拡大表示されますので、囲いたいものとその外との境界付近でクリックして枠を移動させて、境界部分を拡大してください。その後、ウィンドウ内の囲いたいものの内部をクリックすると、その点を端点とする線分が表示されますので、その線分を外側に向けてクリックしてください。運がよければちゃんと囲んでくれるでしょう。もし思いどおりにならない場合はパラメータを適当に変えてみてください（多少の経験が必要です）。また、このウィンドウ上からマスクのペイント（べたマスク）とクリア、および前で説明したMASK. Xを起動することができます。

[ポイント]

図3.26のように現在注目しているピクセルの周囲を1から順に調べ、近似色でなければマスクしてそのピクセルに注目し、同様の作業を繰り返していくことで、近似色の周囲を囲うことができます。

[ソース]wand.c lupe.s
[リンク]EXLIB.L

図3.25 マジックワンド

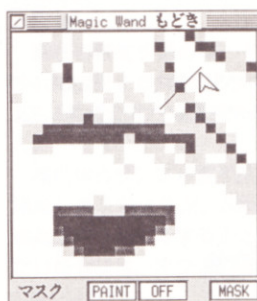
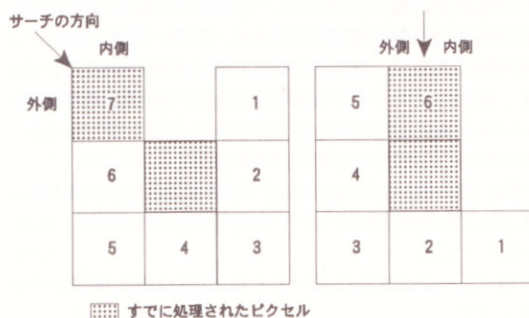


図3.26 境界のサーチ順



3. 4. 4. ぼかし

●MaskShade. x

指定領域内（矩形指定はMaskShade. xで行いますので、矩形指定フラグは必要ありません）のマスクに対して、ぼかしをかけます。アナログマスクを使用できないスモールモデル以下では動作しません。

[ポイント]


任意のピクセルを中心とした3×3領域に、図3. 27のように重み付けを行うことで、ぼかしを行っています。また、マスクをしていない部分は、マスクレベル31とみなして処理を行っています。

[ソース]maskshade. c

[リンク]EXLIB. L

図3. 27 ぼかしの比重

| | | |
|----------------|---------------|----------------|
| $\frac{1}{16}$ | $\frac{1}{8}$ | $\frac{1}{16}$ |
| $\frac{1}{8}$ | $\frac{1}{4}$ | $\frac{1}{8}$ |
| $\frac{1}{16}$ | $\frac{1}{8}$ | $\frac{1}{16}$ |

 ターゲットピクセル

3. 4. 5. 縁取り

●MaskEdge. x [flag] [para:1-9, 4]

指定範囲内のマスクを、よりレベルの高いマスクで縁取りします。パラメータは、縁取り幅を示します。アナログマスクを使用できないスモールモデル以下では動作しません。

[ポイント]

任意のピクセルの周囲8ピクセルを調べ、いちばんレベルの低いマスクと縁取り幅からマスクレベルを演算して、描画を行っています。

[ソース]maskedge. c

[リンク]EXLIB. L

3. 4. 6. 線画マスク変換

●Line2Mask. x

作業画面の線プレーンのレベルが15以下のピクセルを、レベル30のマスクに置き換えます（スモールモデル以下ではべたマスク）。実行後の作業画面のカラーは白になります。線画をスキャナで取り込んだ画像の下準備用の外部ファイルと考えていいでしょう。場合によっては、この前にモノトーンを実行する必要があります。

白い紙に黒いペンなどで描いた線をスキャナで取り込んだことが前提となっていますので注意してください。

[ポイント]

線プレーンの最上位ビットのみに注目して、処理を行っています。

[ソース]Line2Mask. s

3. 5. 3次元階層

この階層は、3次元オブジェクトの描画関連の機能などで構成されています。この階層の描画機能については、かなりのマシンパワーと時間を要します。また、浮動小数点演算を多用していますので、浮動小数点演算プロセッサを搭載することにより、描画速度を上げることができます。

描画機能に関しては、それぞれの形状に対してテクスチャマッピングやバンプマッピングを行うことができますが、パースはかかりません。また、光線は基本的に平行光線のみです。

以下にここで定義されている機能について説明します。

3. 5. 1. ライティング設定

●Light. x

オブジェクトに投射する光に関しての設定を行います。ここで設定されたパラメータは、この階層の描画機能すべてに反映されます。また、それぞれの描画機能から呼び出すこともできます。実行すると、図3. 28のようなウィンドウが開きます。

・直接光強度

光源から直接オブジェクトに投射する光線の強度を示します。

・環境光強度

周囲から均一に投射される光の強度を示します。

・ハイライト強度

投射された直接光が、オブジェクトで反射される光の強さを示します。値を大きくすると、光沢のある質感を与えることができます。

・ハイライト収束率

ハイライトの大きさを示します。値を大きくすると、ハイライトは小さくなり、滑らかな質感を与えることができます。

・平行光ベクトル

投射する光線のベクトルを示します。画面の右方向がX、下方向がY、奥方向がZに対応しています。また、球がサンプル表示されているボックス内でクリックすることでも、ベクトルを設定することができ、右クリックで逆光になります。擬似点光源が設定されているときには、平行光ベクトルは無視されます。

・擬似点光源

それぞれのオブジェクトに対しては平行光線の投射しかできませんが、光源位置を設定することで、それぞれのオブジェクトに光源からオブジェクトの中心に向かう平行光線を投射することができます。設定ボタンをクリックすると、図3. 29のようなウィンドウが開きます。XY位置設定ボックス内に表示されている枠は画面を示し、左上が原点となり、右方向がX、下方向がYに対応しています。その左のスライドバーはZ位置を示し、値が大きいほど奥であることを示します。

・File

現在の設定状態のロード/セーブを行います。3. 5. 6. で説明するスクリプトでLightを指定する場合には、あらかじめここで状態をセーブしておかなければなりません。

[ポイント]

ここで設定されたライティングデータは、Light.xと同じディレクトリにLight.DEFというファイル名で保存され、各描画機能から参照されます。このファイルはテキストファイルで、表3.1のような構成になっていますので、テキストエディタなどで編集することができますが、記述を間違えた場合の動作の保証はできません。また、Fileでセーブした設定状態ファイルも同様です。

[ソース]Light.c ConeLight.c

[リンク]deslib.o EXLIB.L

表3.1 Light.xの設定状態ファイルのフォーマット

| | |
|-------------------------|-------------------|
| VectorX=2.000000 | 平行光Xベクトル |
| VectorY=-3.000000 | Yベクトル |
| VectorZ=4.000000 | Zベクトル |
| Direct=0.700000 | 直接光強度 (0.0~1.0) |
| Environment=0.400000 | 環境光強度 (0.0~1.0) |
| HighlightPower=1.400000 | ハイライト強度 (0.0~2.0) |
| HighlightFocus=8.000000 | 収束率 (0~40) |
| ConeLight=0 | 疑似点光源フラグ(注) |
| ConeLightX=0 | X座標 (-768~1280) |
| ConeLightY=0 | Y座標 (-768~1280) |
| ConeLightZ=0 | Z座標 (0~2048) |

注: 1がON、0がOFF

図3.28 光源設定

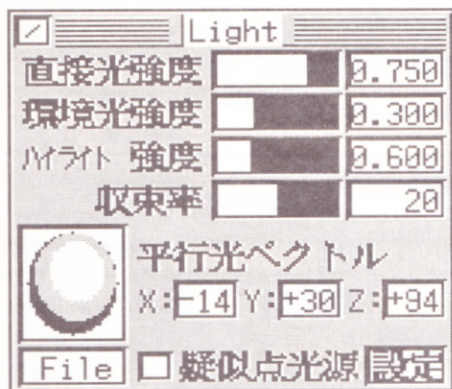
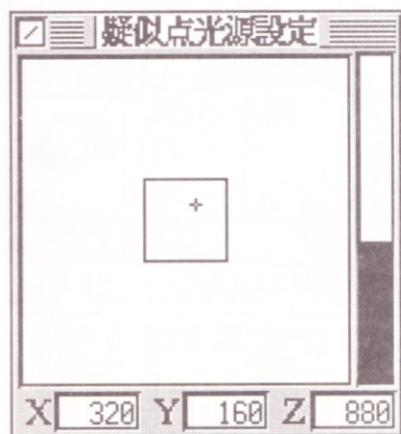


図3.29 疑似点光源



3. 5. 2. 球状変形

●Sphere.x

表示画面に楕円体を描画します。実行すると図3.30のよ

うなウィンドウが開きます。

・X半径

楕円体のX方向の半径を示します。単位は、画面横方向のピクセル幅です。値の変更は、テキストボックスをクリックしてキーボードから入力するか、ラベルを左右クリックして増減させます。ラベルをクリックすると、値は5ずつ変化しますが、シフトキーを併用することで、1ずつ変化させることができます。

・Y半径

楕円体のY方向の半径を示します。単位は、画面縦方向のピクセル幅です。値の変更は、X半径と同様です。

・楕球回転

楕円体をZ軸回りに回転させます。

・ライティング設定

3. 5. 1. で説明したLight.xを起動します。

・Shading

楕円体に陰影をつけます。

・Mapping

テクスチャマッピングとバンプマッピングを行うことができ、実行時にマッピング矩形指定を行うことで、楕球にマッピングすることができます(テクスチャとバンプ両方を指定した場合は、同じマッピング画像を使用することになります)。バンプ右のボックスに示されている数値はバンプの強さを示し、ボックス内でマウスをクリックすることで増減させることができます。矩形指定時にスペースキーを押すことで、裏画面の画像を指定することができ、マッピング画像にアナログマスクを施しておくことで、半透明のオブジェクトを描画することもできます。

・Forward, Back

楕円体の手前半面、奥の半面を描画するかどうか指定します。

・Z位置

ライティング設定で疑似点光源を指定している場合のみ有効で、オブジェクトの中心のZ位置を示します。

・File

現在の設定状態のロード/セーブを行います。3. 5. 6. で説明するスクリプトでSphereを指定する場合には、あらかじめここで状態をセーブしておかなければなりません。

楕円体のサンプル表示は、パラメータを変更するたびに描画し直されますが、SHIFTキーを押しておくことで、描画をキャンセルすることができます。反対に、再描画したい場合には、マウスでクリックしてください。

[ポイント]

マッピングは、中心からの方位と距離が正しくなる、地図でいえば心射図法をとっています。したがって、マッピング画像は矩形で指定しますが、実際には矩形に内接する楕円内しか使用されません。また、マッピングは手前の半球と奥の半球の計2枚行われます。

Fileでセーブした設定状態ファイルはテキストファイルで、表3.2のような構成になっていますので、テキストエディタなどで編集することができます。しかし、記述を間違えた場合の動作の保証はできませんので注意してください。

[ソース]Sphere.c 3DMap.c RevBox.s RevLine.s

[リンク]deslib.o EXLIB.L

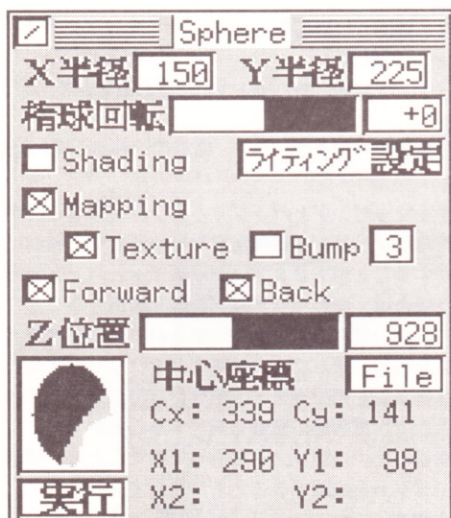
表3.2 Sphere.xの設定状態ファイルのフォーマット

| | |
|-------------|--------------------|
| CenterX=128 | 中心X座標 (0~512) |
| CenterY=128 | 中心Y座標 (0~512) |
| RadiusX=64 | X半径 |
| RadiusY=64 | Y半径 |
| Bark=0 | Z軸周りの回転 (-180~180) |
| PositionZ=0 | 中心Z座標 (0~2048) |
| Shade=1 | 陰影フラグ (注1) |
| Mapping=0 | マッピングフラグ (注1) |
| Texture=1 | テクスチャフラグ (注1) (注2) |
| Bump=0 | バンプフラグ (注1) (注2) |
| Forward=1 | 手前半面描画フラグ (注1) |
| Back=1 | 奥半面描画フラグ (注1) |

注1: 1がON, 0がOFF

注2: テクスチャとバンプの必ずどちらが一方だけがON

図3.30 球状変形

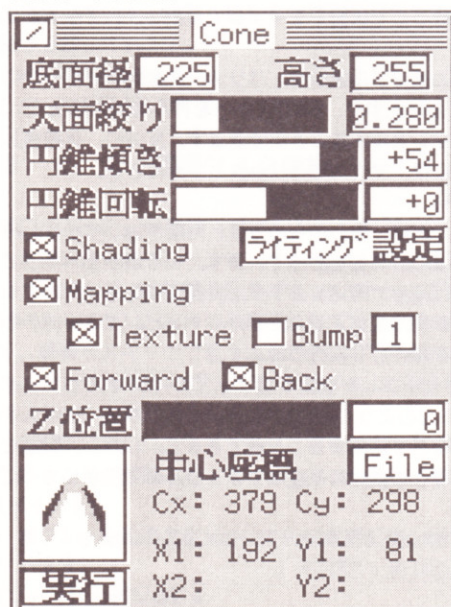


3. 5. 3. 円錐変形

●Cone.x

作業画面に円錐台を描画します。実行すると、図3.31のようなウィンドウが開きます。

図3.31 円錐変形



・底面径

円錐台の底面の直径を示します。単位は、画面横方向のピクセル幅です。値の変更は、球状変形のX半径と同様です。

・高さ

円錐台の高さを示します。単位は、画面縦方向のピクセル幅です。値の変更は、球状変形のX半径と同様です。

・天面絞り

底面径に対して、天面の直径の比率を示します。

・円錐傾き

X軸回りの傾きを示します。

・円錐回転

円錐台をZ軸回りに回転させます。

その他の事項については、3. 5. 2. の球状変形と同様です。

[ポイント]

マッピングは、図3.32のように手前と奥の計2枚行われます。

Fileでセーブした設定状態ファイルはテキストファイルで、表3.3のような構成になっていますので、テキストエディタなどで編集することができますが、記述を間違えた場合の動作の保証はできません。

[ソース]Cone.c 3DMap.c Equation.c RevBox.s RevLine.s

[リンク]deslib.o EXLIB.L

図3-32 円錐へのマッピング

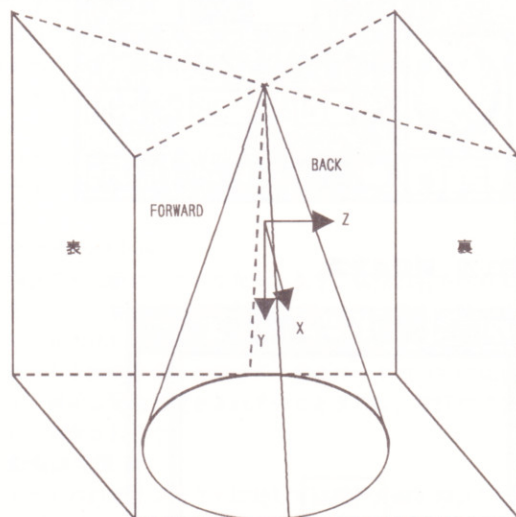


表3.3 Cone.xの設定状態ファイルのフォーマット

| | |
|----------------------|--------------------|
| CenterX=64 | 中心X座標 (0~512) |
| CenterY=64 | 中心Y座標 (0~512) |
| BottomDiameter=128 | 底面径 |
| Height=128 | 高さ |
| TopAperture=0.000000 | 天面径比 (0.0~1.0) |
| Pitch=0 | X軸周りの回転 (-90~90) |
| Bark=0 | Z軸周りの回転 (-180~180) |
| PositionZ=0 | 中心Z座標 (0~2048) |
| Shade=1 | 陰影フラグ (注1) |
| Mapping=0 | マッピングフラグ (注1) |
| Texture=1 | テクスチャフラグ (注1) (注2) |
| Bump=0 | バンプフラグ (注1) (注2) |
| Forward=1 | 手前半面描画フラグ (注1) |
| Back=1 | 奥半面描画フラグ (注1) |

注1: 1がON, 0がOFF

注2: テクスチャとバンプの必ずどちらが一方だけがON

3. 5. 4. 環管変形

●Torus. x

作業画面に環管（ドーナツ型）を描画します。実行すると、図3.33のようなウィンドウが開きます。

・環芯径

環の中心の直径を示します。単位は、画面横方向のピクセル幅です。値の変更は球状変形のX半径と同様です。

・管径

管の直径を示します。単位は、画面横方向のピクセル幅です。値の変更は球状変形のX半径と同様です。

・環管傾き

X軸回りの傾きを示します。

・環管回転

環管をZ軸回りに回転させます。

その他については、3. 5. 2. の球状変形と同様です。

[ポイント]

マッピングは、図3.34のように手前の管の外側、内側、奥の管の外側、内側の計4枚行われます。

Fileでセーブした設定状態ファイルはテキストファイルで表3.4のような構成になっています。ED. Xなどで編集できますが記述を間違えた場合の動作は保証できません。

[ソース]Torus. c 3DMap. c Equation. c RevBox. s RevLine. s

[リンク]deslib. o EXLIB. L

図3.33 環管変形

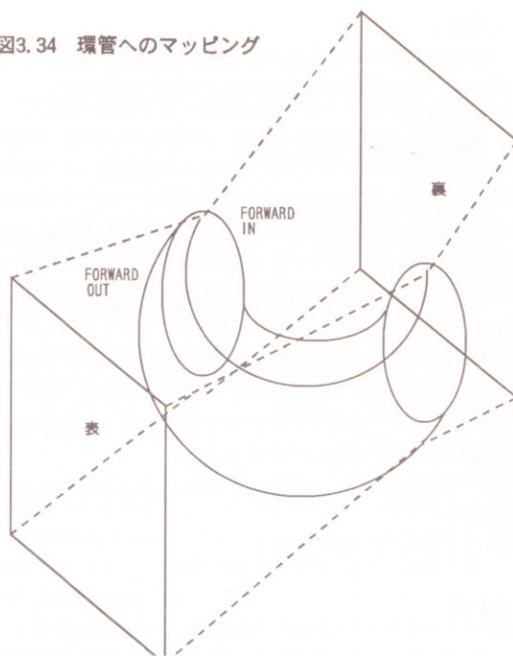
表3.4 Torus. xの設定状態ファイルのフォーマット

| | |
|-----------------|--------------------|
| CenterX=64 | 中心X座標 (0~512) |
| CenterY=64 | 中心Y座標 (0~512) |
| RingDiameter=64 | 環芯径 |
| TubeDiameter=32 | 管径 |
| Pitch=0 | X軸周りの回転 (-90~90) |
| Bark=0 | Z軸周りの回転 (-180~180) |
| PositionZ=0 | 中心Z座標 (0~2048) |
| Shade=1 | 陰影フラグ (注1) |
| Mapping=0 | マッピングフラグ (注1) |
| Texture=1 | テクスチャフラグ (注1) (注2) |
| Bump=0 | バンプフラグ (注1) (注2) |
| ForwardIn=1 | 手前外側描画フラグ (注1) |
| ForwardOut=1 | 手前内側描画フラグ (注1) |
| BackIn=1 | 奥外側描画フラグ (注1) |
| BackOut=1 | 奥内側描画フラグ (注1) |

注1: 1がON、0がOFF

注2: テクスチャとバンプの必ずどちらか一方だけがON

図3.34 環管へのマッピング



3. 5. 5. 放物変形

●Parabola. x

作業画面に回転放物面を描画します。実行すると、図3.35のようなウィンドウが開きます。

・上面径

放物面の上面の直径を示します。単位は画面横方向のピクセル幅です。値の変更は球状変形のX半径と同様です。

・底面径

放物面の底面の直径を示します。単位は画面横方向のピクセル幅です。値の変更は球状変形のX半径と同様です。

図3.35 放物面変形

・高さ

放物面の高さを示します。単位は画面縦方向のピクセル幅です。値の変更は球状変形のX半径と同様です。

・放物傾き

X軸回りの傾きを示します。

・放物回転

放物面をZ軸回りに回転させます。

その他については、3. 5. 2. の球状変形と同様です。

[ポイント]

マッピングは、図3.36のように手前と奥の計2枚行われます。

Fileでセーブした設定状態ファイルはテキストファイルで、表3.5のような構成になっていますので、テキストエディタなどで編集することができますが、記述を間違えた場合の動作の保証はできません。

[ソース]Parabola.c 3DMap.c Equation.c RevBox.s RevLine.s

[リンク]deslib.o EXLIB.L

図3.36

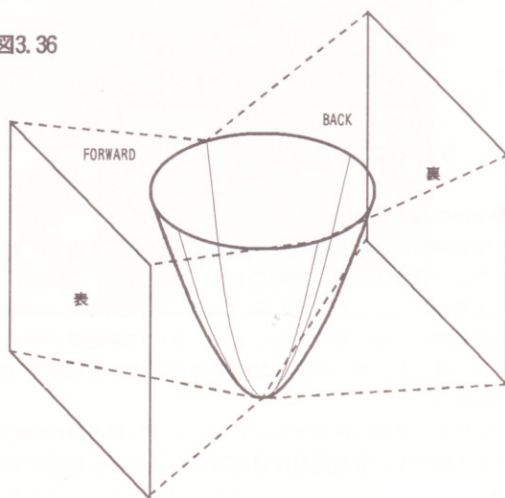


表3.5 Parabola.xの設定状態ファイルのフォーマット

| | |
|------------------|--------------------|
| CenterX=64 | 中心X座標 (0~512) |
| CenterY=64 | 中心Y座標 (0~512) |
| TopDiameter=64 | 上面径 |
| BottomDiameter=0 | 底面径 |
| Height=64 | 高さ |
| Pitch=0 | X軸周りの回転 (-90~90) |
| Bank=0 | Z軸周りの回転 (-180~180) |
| PositionZ=0 | 中心Z座標 (0~2048) |
| Shade=1 | 陰影フラグ (注1) |
| Mapping=0 | マッピングフラグ (注1) |
| Texture=1 | テクスチャフラグ (注1) (注2) |
| Bump=0 | バンプフラグ (注1) (注2) |
| Forward=1 | 手前半面描画フラグ (注1) |
| Back=1 | 奥半面描画フラグ (注1) |

注1: 1がON, 0がOFF

注2: テクスチャとバンプの必ずどちらが一方だけがON

やMappingを再登録するまで有効です。Mapping画像は任意のPICファイルの任意矩形範囲を指定できますが、Mappingを使用した場合には、ワークとして使用しますので、裏画面は破壊されます。登録しない場合には、Lightは現在設定されているライティング設定が有効となり、Mappingは裏画面全体が対象となります。

[ポイント]

描画にはそれぞれの実行ファイルにそれぞれの状態設定ファイルを渡すことで呼び出していますので、空きメモリが足りないときには、途中で異常終了する場合があります。

Fileでセーブした設定状態ファイルはテキストファイルで、表3.6のような構成になっていますので、テキストエディタなどで編集することができますが、記述を間違えた場合の動作の保証はできません。

[ソース]Script.c xpic.s RevBox.s RevLine.s

[リンク]deslib.o EXLIB.L

図3.37 3Dスクリプト

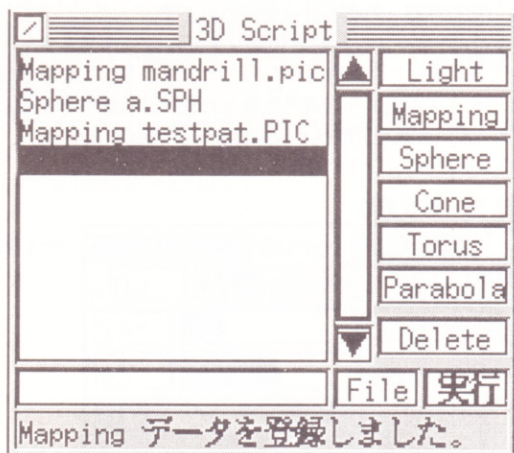


表3.6 Script.xの設定状態ファイルのフォーマット

```
Light B:\3DMap\default.LIG
Mapping B:\BACK\BB2.PIC 0 0 511 511
Sphere B:\3DMap\default.SPH
Cone B:\3DMap\default.CON
Torus B:\3DMap\default.TOR
Parabola B:\3DMap\default.PAR
```

注: ファイル名はフルパスで指定、Mappingの場合のみ範囲を指定する

3. 6. サンプル階層

この階層は、第4章「外部ファイルの作成」で外部ファイル作成例として説明されていますので、そちらを参照してください。

3. 5. 6. スクリプト

●Script.x

あらかじめセーブしておいた3D描画関係の状態ファイルを登録することで、一括実行することができます。実行すると、図3.37のようなウィンドウが開きます。

Light~Parabolaボタンをクリックして、状態ファイルを順次登録してください。LightやMappingは描画したいオブジェクトの前で登録し、一度登録すると、次にLight

4. 外部ファイルの作成

菊地 功

本書にはEX-System専用外部ファイル作成用のC言語のライブラリとアセンブラのマクロファイルが収録されていますが、ここではC言語で作成法を説明していきます。なお、以降はC言語の知識があり、開発ツールおよび環境が整っているものとして説明を進めます。以下は筆者が使用しているツールです。

- C compiler ver. 2 のライブラリおよびMAKE
SHARP/Hudson
- GNU C compiler 真理子バージョン ver. 1.20
Free Software Foundation, Inc.
- High-speed Assembler ver. 3.09
中村祐一氏
- SILK Hi-speed Linker ver. 3.01
そと氏

代用品の場合はそれに応じて対応が必要な場合があります。それらについては個人の責任で行ってください。

ここで説明されている外部ファイルのソースは、Source.Lzhをディレクトリ付きで解凍したあとのEffectの中にはほかの外部ファイルと一緒に収録されています。

4. 1. 環境設定

インクルードファイルおよびライブラリをハードディスクにコピーします。EXLIB.HとEX.Hを環境変数includeに指定されたディレクトリへ、EXLIB.Lを環境変数libに指定されたディレクトリへコピーしてください。アセンブラを使用される方は、マクロファイルEXCALL.MACを環境変数includeに指定されたディレクトリへコピーします。また、C言語およびアセンブラのオンラインライブラリマニュアルEXLIB.DOCとEXCALL.DOCも、どこか手近なところへコピーしておくとういでしょう。

環境変数の設定は、個々の環境によって違いますが、念のため筆者の開発に関係する環境変数を表4.1に示します。環境変数の意味についてはそれぞれのツールのドキュメントを参照してください。注意が必要なのは、GCCが参照する真理子などの全角の環境変数は、XCのMAKEを通りません。

GCCでコンパイルの際には、-fno-defer-popオプションを指定して、スタック一括補正を禁止してください。このオプションを指定しないと、ユーザーモードとスーパーバイザモードを行き来する際に、スタックポインタが狂ってバスエラーなどを起こすことがあります。

表4.1 環境変数設定例

```
lib=b:\lib
include=b:\include
MARINO= ABD
MARINA= ed
GCC_OPTION= FMLOE+
GCC_LIB= l
GCC_AS=HAS
GCC_LINK=HLK
```

4. 2. 画面の構成

VRAMの構造には一般的に2つあり、ひとつはPackedVRAM、俗にいう垂直型、もうひとつはPlannedVRAM、水平型です。垂直型は1回のメモリアクセスで任意のピクセルのカラーコードを拾うことができますが、水平型はプレーンに分解されており、それぞれのプレーンのピクセルビットを合わせたものがカラーコードになります。X680x0のテキストVRAM（\$E00000から512Kバイト）は水平型ですが、グラフィックVRAM（\$C00000から512Kバイト、以下G-RAMとする）は垂直型になります。G-RAMは16ビットで1ピクセルを表しますので、通常unsigned shortのポインタでアクセスします。この16ビットのうち、65536色モードでは、上位ビットからG5ビット、R5ビット、B5ビット、I1ビットという構成になっています。Iというのは輝度ビットと呼ばれ、全体の輝度を上げる働きをしますが、EX-Systemではこのビットはマスクの存在情報として使用されますので、実際に使用できるのはRGB各5ビットの32768色になります。

EX-Systemには、表示画面であるG-RAMのほかには画面関連で待避画面、裏画面、マスクバッファを確保しています。待避画面とは、基本的に表示画面と同じ内容ですが、ウィンドウの下や、マスクの下の情報も保持しています。表示画面では、マスク部分は上位5ビットがマスクレベル、最下位ビットが1で表されているのに対して、待避画面はマスク下のカラーコードに最下位ビットを立てたデータで保管されており、裏画面も待避画面と同じ構造をしています。マスクバッファは表と裏があり、それぞれ1ピクセル8バイトで確保され、上位5ビットにマスクレベルを保持しています。

4. 3. 外部ファイルへの引数と戻り値

EX-Systemから外部ファイルを実行する場合、いくつかの引数を与えられて呼び出されます。なにが与えられるかはコンフィグファイルの記述で決まり、以下のような順で指定されます。

ExecFile another [x1 y1 x2 y2] [para1 [para2]] [option]

• another

Z's-EX ver. 1.1で裏画面アドレスを得るために与えられた引数ですが、ver. 2.0以降ではEXコールで得ることができます。互換性のための引数で、すべての外部ファイルに渡されます。

• x1 y1 x2 y2

実行時にシステム側で矩形指定を行った場合に与えられる矩形範囲です。(x1, y1)が矩形左上座標、(x2, y2)が右下座標を示し、10進の文字列で与えられます。

• para1, para2

パラメータが指定された場合に与えられるパラメータの値で、10進の文字列で与えられます。

• option

オプションが指定された場合に与えられるオプション文字列です。

外部ファイルが終了し、システムに処理が戻ってきたときに、外部ファイルの戻り値によって、システムが以下の

ような処理を行います。

- 0…表示画面から待避画面にコピーする
- 1…待避画面から表示画面にコピーする
- 2…なにもしない
- 3…待避画面のマスクを考慮して表示画面と合成する
- 1…EX-Systemを終了する

その他…エラーメッセージを表示し、待避画面から表示画面にコピーする

0や1は以前のバージョンからの互換性のためと考えて差し支えありません。共に同じ動作をするEXコールを用意してありますし、表示画面から待避画面にコピーする場合には、戻り値3か4。4。1で紹介するCOMGVRAM()のほうを使用することをおすすめします。また、—1はESCキーを押した場合などに使用されます。

戻り値3とCOMGVRAM()は同じ機能を果たしますが、戻り値3で終了した場合には機能するタイミングが特殊です。外部ファイルを単独で実行した場合には、ほかの場合と変わりませんが、連続実行した場合には、2つの外部ファイルの実行が終了した時点で処理が行われます。これは、ひとつ目の外部ファイルが処理した結果を、待避画面との合成前に2つ目の外部ファイルに渡したい場合を考慮してのことで、今回収録された外部ファイル中では、Monotoneと併用するDifferなどがそれに相当します。Monotoneは、処理を表示画面にだけ書き込み、戻り値3で終了します。単独実行であればそこで作業画面と合成されますし、Differとの連続実行であれば、Differは表示画面に対して処理を行えばいいことになります。ここで、もしDifferが待避画面を見て処理してしまったら、Monotoneが施される以前の画像を参照することとなり、正しく処理を行うことができません。つまり、アナログマスクに対応した外部ファイルを連続実行させる場合には、連続実行されることを考慮して作成された外部ファイルでなければ、思ったような結果が得られないことがあるというわけです。

4. 4. 画面フィルタの作成

まずは簡単な画面フィルタを作ってみましょう。

4. 4. 1. メディアンフィルタ (Median.c)

ターゲットピクセルの周囲の色を調べ、明るさの順で真ん中の色をターゲットピクセルの色とします。自然画などのノイズ除去に有効です。ここではターゲットピクセルを中心とした3×3のマトリクスで明るさを調べ、5番目に明るい色でペイントしていきます。明るさは、HSV変換や、RGBそれぞれで調べていたのでは時間がかかりますので、RGBの合計で行うことにします。処理領域はシステム側の呼び出し時に行ってもらうことにして、プログラムに取り掛かりましょう。

まずEXLIB.Hをインクルードしてください。これはEXコールを使用する場合は必ず指定しなければなりません。

main()関数の先頭では、矩形範囲を引数から拾っています。矩形指定を行った場合には引数は6つ以上（実行ファイル名含む）、行わない場合は5つ以下です（オプションにスペースが含まれていなければ）、矩形指定を行ったかどうか引数の数から判断できます。矩形指定を行わなかった場合には、変数の初期値で画面全体を対象とする

ようになっています。

次に本体のMedian()関数ですが、スーパーバイザ領域に直接アクセスしますので、まずスーパーバイザモードに移行しています。スーパーバイザ領域とは、G-RAMなどのハードウェアのシステム領域のことで、スーパーバイザモードに移行せずにこの領域にアクセスすると、バスエラーが発生します。また、Z's-EXの待避画面はテキストVRAMを使用していますので、待避画面にアクセスする場合にもスーパーバイザモードでなければなりません。

G-RAMのアクセス方法は前項で説明しましたが、カラーコードを調べるのに待避画面のアドレスを得る必要があります。これはEXコール\$1FのBUFFADR()の戻り値で得ることができます。構造は表示画面と同じですので、同じようにunsigned shortのポインタでアクセスすればよいでしょう。

さて、Median()の処理内容ですが、同じピクセルを2度読みしないように、バッファを使用しています。配列の1番目のサフィックスは0が（最下位ビットを除いた）カラーコード、1がRGBの合計、2番目がY座標の3の剰余、3番目がX座標を示し、画面上方からスキャンしています。バッファ内から、ターゲット近傍の9ピクセルを拾い、5番目をピックアップするのですが、標準関数のqsort()があまり速くなかったので、ループを回して調べることにしました。最後にそれで得られたカラーコードを表示画面に書き込み、表示画面と待避画面のポインタを次ラインに進めています。

Median()関数から帰ってきてから、COMGVRAM()という関数を読んでいます。これは表示画面に書き込んだ内容を、適当に待避画面とマスクバッファに反映させるというEXコールです。具体的には同じ座標のピクセルが、

- ・表示画面、待避画面ともにマスクされていない（最下位ビットが立っていない）場合

表示画面から待避画面にコピーする。

- ・表示画面にマスクされていた場合（待避画面に関係なく）

上位8ビットをマスクバッファに転送する。ただし、マスクレベルが\$1 (\$F801) の場合はマスクをクリアする。

- ・待避画面だけにマスクされていた場合

マスクのレベルを考慮して表示画面カラーを待避画面に書き込む。

この関数を使用することで、多くの場合はほとんどマスクを意識することなくプログラムを作成することができます。ただ、カラーとマスクを同時に更新できないという欠点があります。

最後に外部ファイル側で待避画面への転送を行ったので、戻り値を2として終了します。ここで、COMGVRAM()を呼ばずに、戻り値3で終了しても同じ結果を得ることができます。このプログラムのように、待避画面から参照して表示画面に書き込み、最後にCOMGVRAM()で待避画面に転送するというのが、いちばん安全かつ簡単な方法でしょう。待避画面を書き換えなければ、途中でシステムエラーが発生した場合や、インタラプトを押した場合でも、システム側が画像を再生してくれます。

コンパイルには、EXLIB.Lを忘れずにリンクしてください。また、GCCでは、4。1でも述べましたが、-fno-defer-popオプションを必ず付けてコンパイルしてください。また、ほかの外部ファイルと一緒にメイクファイルに

登録してありますので、そちらを使う場合には、
make median. x
と実行してください。

4. 4. 2. 表裏加算 (AddScr. c)

裏画面を作業画面に加算します。カラーをRGBの3要素に分解し、それぞれ加算しますが、ここではマスクも加算することにしましょう。ただし、マスクレベルをそのまま加算すると、どんどん薄くなってしまいます。マスクは加算することで濃くなってほしいので、マスクレベルを31から減じた値で加算し、最後にまた31から引いてやることにします。また、マスクの値も操作しますので、カラーはマスクの有無に関らず処理を行うことにしましょう。さらに、符号を変えるだけで減算にすることもできますので、オプションに「-」が指定された場合は表裏減算を行うようにします。

メディアンフィルタと同様にシステムで矩形指定を行わせることにしますが、今度はCOMGVRAM()を使えませんが、マスクバッファにも自分で書き込んでやらなければなりません。AddScr()関数内でメディアンフィルタ同様にスーパーバイザモードに移行し、G-RAMへのポインタと待避画面へのアドレスを得ていますが、さらに今度は裏画面とマスクバッファのアドレスが必要です。裏画面のアドレスはEXコール\$00のGETADR()の戻り値で、マスクバッファはMASKADR()に引数0および1を渡してやることで、表および裏のアドレスを得ることができます。これらの関数は、モデルサイズによりバッファが確保されていない場合はNULLが返ります。この外部ファイルの機能からいって、裏画面がない場合はどうしようもないので、エラーメッセージを出して終了することにします。main()関数に戻ってEXコール\$01のCONFIRM()関数で確認ウィンドウを表示しますが、外部ファイルが呼び出された時点ではマウス形状が時計のままであるので、EXコール\$09のMCSET()関数に引数0を与えることで通常状態に戻しておきましょう。

さて本処理ですが、ループ内で場合分けをすると遅くなってしまうので、アナログマスクが使用できる場合と使用できない場合とでループを2つ作りました。前のループが使用できる場合、後ろが使用できない場合ですが、後ろは前のループからアナログマスク処理を省いたものですので、使用できる場合のループに沿って説明しましょう。カラーについては先ほど説明したように、RGBそれぞれを加算または減算してまた結合しただけですが、マスクがあるかどうかで最下位ビットが変わってきます。マスクは31から減じた値で加算または減算しますが、マスクのない部分はレベル31、すなわち31から減じて0として演算しています。ただし、マスクレベルは8ビットのうち上位5ビットで、特にビットシフトの必要もなかったため、3ビットの下駄を履かせたまま演算しています。

表示画面は、マスクがなかった場合はカラーコードを、マスクがあった場合は上位にマスクレベルを入れ、最下位ビットを立てています。待避画面とマスクバッファを更新しましたので、最後にEXコール\$05のBUF2GRAM()を使うか、戻り値を1としてプログラムを終了してもいいのですが、実行経過が見えないというのも不安なので、同時に書き換えるようにしました。また、今回は裏画面を参照し

て表画面を書き換えたのですが、アクセスできるからといって裏画面や裏マスクを書き換えることは、礼儀として極力避けてください。

4. 4. 3. 裏→表転送 (Another2Gram. c)

裏画面を作業画面にコピーします。その際に、裏マスクは裏画面のカラーを見にくくし、表マスクは作業画面に書きにくくしてしましましょう。ただ、それだけではシステムで裏画面カラーのボックスフィルを行った場合と同じですので、パラメータの拾い方を兼ねて裏画面に任意レベルのマスクを重ねた処理を加えてみます。これでアナログマスク対応の表裏合成と同じ結果を得ることがができます。パラメータは0~7として、31からパラメータを4倍した値を引いたレベルのマスク（パラメータが大きいほどマスクが強い）を重ねるようにし、裏マスクとパラメータによるマスクの乗算により、新しいマスクレベルを算出します。具体的には、裏マスクをmask、パラメータをparaとすると、新しいマスクレベルは、

$$\text{mask} * (31 - \text{para} * 4) / 31$$

で表されることになります。

今回は矩形指定はシステムで行うようにし、裏画面が使用できない場合はエラー終了しますが、アナログマスクが使用できない場合も終了するようにしました。アナログマスクが使用できない場合はCompose. xと同等ですから。本処理では、先ほどの式で新しい裏マスクを算出していますが、ここでもマスクのない部分はレベル31で演算しています。

さて、そのあとですが、表と裏でマスクがある場合とない場合で場合分けをしなければなりません。作業画面カラーをbuf、裏画面カラーをano、表マスクレベルをmsk0、(新しい)裏マスクレベルをmsk1とすると、新しい作業画面カラーは次の式で表されます。

・表裏どちらにもマスクがない場合

$$\text{buf} = \text{ano}$$

・表だけにマスクがある場合

$$\text{buf} = (\text{buf} * (31 - \text{msk0}) + \text{ano} * \text{msk0}) / 31$$

・裏だけにマスクがある場合

$$\text{buf} = (\text{buf} * (31 - \text{msk1}) + \text{ano} * \text{msk1}) / 31$$

・表裏両方にマスクがある場合

$$\text{buf} =$$

$$(\text{buf} * (31 * 31 - \text{msk0} * \text{msk1}) + \text{ano} * \text{msk0} * \text{msk1}) / (31 * 31)$$

RGBそれぞれについてこの演算を行わなければなりません。いちばん上以外はSource. Lzhを解凍したときのSystem¥Draw¥Draw. hでマクロに登録してあります。ただし、除算を行うと遅くなりますので、正確ではありませんがシフトを用いています。

・表だけにマスクがある場合

[マクロ] PointSetWithMask0(buf, msk0, ano)

[演算式] $\text{buf} = (\text{buf} * (32 - \text{msk0}) + \text{ano} * \text{msk0}) \gg 5$

・裏だけにマスクがある場合

[マクロ] PointSetWithMask1(buf, ano, msk1)

[演算式] $\text{buf} = (\text{buf} * (32 - \text{msk1}) + \text{ano} * \text{msk1}) \gg 5$

・表裏両方にマスクがある場合

[マクロ] PointSetWithMask01(buf, msk0, ano, msk1)

[演算式] $\text{buf} = ((\text{buf} \ll 10) + (\text{ano} - \text{buf}) * \text{msk0} * \text{msk1}) \gg 10$

演算式はイメージですので、実際にはRGBがパックされた形式でマクロに渡せば、マクロ内部でRGBについて

それぞれ演算され、バックされたあとにbufに代入されます。ただし、マスクレベルはマスクバッファのイメージではなく、3ビットシフトダウンした5ビットを使用してください。また、このマクロはC言語でしか使えませんが、アセンブラでも同様の演算を行う関数をSystem\$Draw\$Draw.sで定義してあります。

Draw.hをインクルードしておけば、面倒な演算をしなくてもこれらのマクロを呼ぶだけ済むのですが、あくまでもマクロですので、インクリメンタルポインタなどを渡さないようにしてください。

あとは、表画面にマスクがなかった場合には、忘れずに表示画面も描画してやれば、戻り値を2としてプログラムを終了していいでしょう。

4. 5. ウィンドウを開く外部ファイルの作成

EX-Systemには、ウィンドウ操作をサポートするEXコールも用意されています。また、吉田泉氏によるウィンドウデザイナを使用することで、より手軽にウィンドウを作成することができます。ここでは、ウィンドウデザイナで作成したウィンドウスケルトンに手を加えることで、ウィンドウを使用する外部ファイルを作成します。ウィンドウデザイナの使用法などについては、そちらの説明を参照してください。

4. 5. 1. 情報表示 (About.c)

EX-Systemの名称、バージョン、それぞれのバッファのアドレスなどを、ウィンドウに表示します。ただ、表示させるだけでは面白くないので、ビューポートを用いて上下にスクロールさせることにします。

まず、ウィンドウデザイナで図4.1のようなウィンドウを作成してください (About.des)。左の大きなボックスがビューポートで、カスタムコントロールを使います。右には上下のスクロールボタンと、ビューポートの位置を表示するためのスライダーを、これもカスタムコントロールでデザインします。とりえずこれだけコントロールを配置したら、About.cという名前でのスケルトンを作成しましょう。

作成されたスケルトンは、特に手を加えなくても、基本的なウィンドウ操作はできますので、ここで一度コンパイルして動かしてみてもいいかもしれません。この時点で操作するのは、ウィンドウと各コントロールの表示、タイトルバーのドラッグによるウィンドウ移動、クローズボックスによるウィンドウクローズ (終了) と、ESCキーによるEX-Systemの終了です。これにコントロールの中身の描画と、コントロールをクリックされたときの動作を付け加えていくことになります。

ソースを見ると、ITEM型のitem[]という配列に数値の羅列が入っています。これらはウィンドウ上に表示されるコントロールなどのアイテムの情報で、クリックされたときのシステム側の動作や、アイテムの座標などが格納されています。ひとつのアイテムについて10個の数値が対応しており、ソース中では1行がアイテムひとつ分になります。その10個の数値の意味は順に、

・反転/クローズ/移動

ビット0がマウスがアイテム上にあるときに反転するか

どうかのフラグを示します。また、254のときはウィンドウ移動 (システム側で移動を行う)、255のときはウィンドウクローズ (特に意味はない) を示します。

・復帰タイミング

アイテム上でマウスがクリックされたときに、ボタンを放すまで待つかどうかを示します。0で待たない、1で待ち、2のときはダブルクリックやトリプルクリックまで検出します。2では多少反応が遅れますので、ダブルクリックなどを検出する必要がない場合には、1にするとよいでしょう。

・アイテム座標 (4個)

ウィンドウ左上からのアイテムの相対座標です。順に左上X、Y座標、右下X、Y座標を示し、マウスカーソルがこの矩形内にあるとき、マウスがアイテム上にあると判断します。

・反転領域 (4個)

ウィンドウ左上からの反転領域の相対座標で、マウスがアイテム上にあるときにこの矩形が反転表示されます。順に左上X、Y座標、右下X、Y座標を示しますが、最初の数値のビット0が0のときは意味をなしません。

この配列のアドレスは、さらにITEMPTR型のitemptrという構造体のメンバになり、システムに伝えられます。ITEMPTRはウィンドウ全体の情報を保持しており、以下のメンバで構成されています。

```
struct ITEMPTR {
    int    x0;      ウィンドウ左上X座標
    int    y0;      ウィンドウ左上Y座標
    int    x;       ウィンドウXサイズ
    int    y;       ウィンドウYサイズ
    ITEM   *i;      アイテムへのポインタ
}
```

これらの情報はウィンドウが表示されているあいだはずっとシステム側から参照され、システムでウィンドウ移動を行った場合にはx0とy0が書き換えられますので、解放してしまわないように注意してください。グローバルで定義しておけば安全でしょう。

ウィンドウ情報をシステム渡すには、EXコール\$10のWINITEM()の引数にアドレスを指定すればいいのですが、ウィンドウデザイナを使用した場合には、デザイナ側の関数set_window_item()内でやってくれています。その後、EXコール\$11のOPENWIN()により、渡しておいたウィンドウ情報に従ってウィンドウを表示します。ただし、これだけではタイトルバーとクローズボックスだけののっぺらぼうのウィンドウしか表示されませんので、デザイナの関数show_all_ctrl()で各コントロールを表示しています。

ウィンドウが表示されたら、マウスがアイテム上にあるか、ボタンは押されたか、などのチェックが必要ですが、これはEXコール\$12のMANAGE()がすべてやってくれます。引数はint型で、ビット0がマウスクリックによりアイテムを選択するまで待つかどうかのフラグを、ビット1がスペースキーによる表裏画面反転、XF1キーによるマスク透視、およびUNDOキーによるアンドゥを許すかどうかのフラグを示します。デザイナのデフォルトでは1、すなわちアイテムが選択されるまで待ち、表裏画面反転などは許さない設定になっています。戻り値もint型で、マウスのボタン、選択されたアイテム番号などを下位12ビットに以下のような構造でバックされています。

戻り値 F L R L R N N N N N N N

・F

マウスがアイテム上になくときのフラグで、引数のビット0が0のときのみ有効です。

・LL

左ボタンの状態を示し、ビットを繋げた2進数で押ししていない状態からトリプルクリックまでを表します。

・RR

右ボタンの状態を、左ボタンと同様に表します。

・NNNNNNNN

アイテム番号を示します。255のときはESCキーを示し、表裏反転が行われたときは254、マスク透視およびアンドゥが行われたときは253を返します。

この戻り値を場合分けすることで、それぞれのコントロールに対応した処理に振り分けることができます。ESCキーが押されたときや、クローズボックスを選択した場合はEXコール\$14のCLOSEWIN()でウィンドウを閉じて終了しますが、ESCキーの場合は終了コード-1を返してEX-Systemも終了するようになっています。

では、いよいよ処理を付け足していくことにしましょう。と、その前に変更する点を挙げてみましょう。まずITEM[]配列のひとつ目と2つ目のフラグです。ID0のカスタムコントロールは、情報を表示するだけのボックスでしたから、反転もボタンを放すのを待つこともしません。ID1と2のスクロールボタンは、反転はしますが、押しているあいだはスクロールし続けてほしいので、これも放すのを待たないことにします。ID3はカスタムコントロールで作ったスライダーで、クリックすることでビューポートを移動するようにしますが、反転はしませんし、ドラッグできるように放すのを待たないことにします。ID4はウィンドウ内のコントロールがない部分、ID5はウィンドウの外を示していますが、なにもさせないので、どちらのフラグも0にしておきましょう。次にMANAGE()の引数ですが、アイテムが選択されるまで待ち、表裏反転なども許して置いて差し支えないでしょう。

変更はこれくらいですので、今度は付け足していくことにします。まず表示する情報を保持する文字配列と、文字列の行数、さらにビューポートに表示されているいちばん上の行番号を保持する変数を、それぞれStrItem、StrLine、CurLineという名前でグローバル宣言しておきましょう。そして、これらの内容を設定すると同時に、コントロール内に描画も行いますので、set_all_ctrl()のあとにInitWindow()という関数を設けて、そこで初期処理を行うことにします。EX-SystemのタイプはEXコール\$20のEXNUM()で調べることができますし、バージョンは、EXコール\$17のVERSION()で得ることができます。

あとはバッファですが、それぞれのバッファに対応した関数の戻り値からアドレスを取得し、戻り値がNULLの場合は確保されていないことを示します。文字列の設定がきたら、ShowStr()関数でビューポートに文字列を表示します。表示座標は、反転させてはいませんが、反転領域座標にコントロールの中身の座標を保持していますので、それから算出しています。

また、EXLIB.Hでは、ウィンドウで使用されている色を次のマクロで定義してあります。

WHITE \$FFFE 白
GRAY1 \$B5AC 明るい灰

GRAY2 \$6318 暗い灰

BLACK \$0000 黒

文字列を表示したら、次はスライダーをShowSlider()関数で描画します。スライダーは反転色で描画しておけば、ビューポートを移動させるときには、もう一度同じところを描画させることでスライダーを消すことができます。反転色でボックスを描画するには、EXコール\$07のREVBOX()を使います。これでウィンドウの初期化は終了しました。

次はコントロールをクリックされたときの処理を書いていきます。必要なのは、ID2、3のスクロールボタンと、ID4のスライダーです。スクロールボタンを押された場合は、ビューポート内を1行スクロールさせて、新たに現れた行を描画するという作業になりますが、矩形をスクロールアップ/ダウンさせるのに、EXコール\$0EのROLLUP()および\$0FのROLLDOWN()という関数が用意されています。ROLLUP()、ROLLDOWN()にはROLLPTR構造体を使用し、以下のようなメンバで構成されています。

```
struct ROLLPTR {  
    int      x0;    領域左上X座標  
    int      y0;    領域左上Y座標  
    int      x;     領域Xサイズ  
    int      y;     領域Yサイズ  
    int      d;     スクロールドット数  
    ushort   c;     下地カラー  
}
```

この関数は、いちばん上または下の行の塗り潰しもやってくれますので、あとは新しい行をsymbol()で描画するだけです。スクロールボタンを押されたときに呼ばれる空の関数button_01()とbutton_02()はデザイナーが宣言してくれているので、その中に記述していきましょう。

最後にスライダーのcustom_ctr_02()ですが、マウスの座標から新たなCurLineを算出し、すでに作成してある関数ShowSlider()とShowStr()関数で、スライダーと文字列を書き直すだけです。

これで完成です。コンパイル時には、デザイナーのライブラリdeslib.oをリンクすること、Draw.hをカレントに置いておくことを忘れないでください。コンフィグファイルに登録して実行してみると、図4.2のようなウィンドウが開くはずですが、ちゃんとビューポートがスクロールするか確かめてみてください。

4. 5. 2. ルーベ (PointLupe.c)

ウィンドウ内にルーベを開き、作業画面を部分拡大して見ることができます。システムのルーベと比べて特別なことができるわけではありませんが、EXコールの説明のために作成してみます。ルーベは2倍、4倍、8倍、16倍の4段階に拡大率を変えることができ、右クリックでスポイトした色を、左クリックでルーベ内またはウィンドウ外に描画できるようにしましょう。EXコールには、ルーベを扱う関数がいくつか用意されていますので、それらを用いることで簡単にルーベを作成することができます。

まずはデザイナーで図4.3のようなウィンドウを作成してください(Lupr.des)。真ん中の大きなカスタムコントロールがルーベボックス、その上下左右の細長いボタンがスクロールボタンです。左下の4つ並んだボタンが倍率ボ

タン、その右にカレントカラーを表示するボックスをカスタムコントロールで作成しましょう。さらに、右下のスペースには、マウスの座標を表示してみます。

では、ソースをいじっていきましょう。まずはitem[]配列です。今回はウィンドウの移動をプログラム側で行うことにしましょう。というのは、ウィンドウの被拡大領域を反転色で描画するのですが、ウィンドウを移動させるときにこのボックスを消しておかないと、ウィンドウにかぶった部分が欠けてしまうからです。そこでタイトルバーを示す2番目のアイテムの最初のフラグを0にしておきます。あとは特に説明しませんが、機能に合わせて適当にフラグを書き換えておきます。さて、main()関数では、About.cと同様にshow_all_ctrl()関数のあとでInitWindow()関数でウィンドウを初期化します。この関数内では、まず各バッファのアドレスを拾ったあと、ルーベの表示を行っています。ルーベを表示するには、EXコール\$27のSETLUPEWINDOW()でルーベ情報をシステムに渡したあと、EXコール\$29のPAINTLUPE()をコールします。SETLUPEWINDOW()の引数は2つで、ひとつ目はルーベのナンバー、2つ目はルーベ情報構造体へのアドレスです。EX-Systemでは、同時に2つまでのルーベを表示することができ、0と1のナンバーで管理します。ここではルーベをひとつしか表示しませんので、ナンバー0を使用します(ナンバー1のみを使用することはできません)。ルーベ情報構造体は以下のメンバから構成されています。

```
sLupeWindow {
    int    x1;    被拡大領域左上X座標
    int    y1;    被拡大領域左上Y座標
    int    x2;    被拡大領域右下X座標
    int    y2;    被拡大領域右下Y座標
    int    wx1;   拡大領域左上X座標
    int    wy1;   拡大領域左上Y座標
    int    wx2;   拡大領域右下X座標
    int    wy2;   拡大領域右下Y座標
    int    f;     拡大率
}
```

これらのメンバには、 $(x2-x1+1)*f=wx2-wx1+1$ 、 $(y2-y1+1)*f=wy2-wy1+1$ の関係が成り立ちます。この構造体を登録しておき、PAINTLUPE()の引数でルーベナンバーを指定することで、ルーベ内を描画することができます。

ルーベの描画ができたので、次にLupeBox()関数で被拡大領域を表示画面に反転色で描画します。ここではREVBOX()ではなく、RevBox()を使いましょう。これはEXコールではなく、RevBox.sに記述してある関数で、ウィンドウを避けて反転色のボックスを描画してくれます。

また、ここでは使用しませんが、ウィンドウを避けて反転色で描画する関数として、RevLine.sで記述してある直線を引く関数RevLine()、RevEllipse.sで記述してある楕円を描画する関数RevEllipse()を用意してあります。ただしこれらの関数は、あらかじめスーパーバイザモードに移行しておいてからコールしてください。その後は、EXコール\$08のREVFILL()で現在の倍率を示すボタンの反転(起動時は2倍)、カレントカラーの表示(起動時は黒)などを行っています。

ウィンドウの初期化が終了したら、メインループです。今度はマウスの座標を逐一表示するために、MANAGE()の引数を2にします。今回はESCキーを押されたときの

処理を最初にやっておくことにしましょう。ルーベボックスを消したあと、SETLUPEWINDOW()のルーベナンバー0にルーベ情報構造体アドレスとしてNULLを渡すことで、リセットを行っています。ルーベを使用したあとは、必ずこの操作を行ってリセットするようにしてください。CLOSEWIN()とreturn()は前回同様です。

次に、表裏反転やマウス透視を行った場合には、ルーベボックスが消えてしまっていますので、再描画することになります。そのあとにマウスの座標を表示させるのですが、ルーベの中にマウスがある場合はルーベ内座標を実座標に変換して表示すべきです。これにはEXコール\$2AのLUPE2SCREEN()を利用します。この関数は、X座標とY座標を格納するint型変数のアドレスを渡してやると、ルーベ上であればその座標をルーベ内座標とみなして実座標に変換しますが、ルーベ外であればなにも行いません。したがって、この関数に座標を通すだけで、ルーベ上であるかどうかを意識せずに実座標を得ることができます。実座標を拾ったら、Dig1()関数で座標の表示を行います。ここでは単に特定座標に数値を表示しているだけです。

ESCキーが押された場合の処理は最初に行ったため、これ以降はマウスがクリックされた場合の処理だけです。ここでクリックされていない場合を弾いておくといでしょう。MS_RIGHTやMS_LEFTはEXLIB.Hで定義されており、MANAGE()の戻り値とアンドを取ることで、それぞれ右ボタンが押されたか、左ボタンが押されたかを調べることができます。

また、ほかにもMS_SINGLE、MS_DOUBLEが宣言されています。今回はウィンドウの移動をプログラムで行うことにしていました。しかし、ウィンドウを移動させるには、EXコール\$13のMOVEWIN()を使用すれば簡単です。引数として現在のマウス座標を渡すだけです。被拡大領域を示すボックスを消去したあとにMOVEWIN()でウィンドウを移動させてください。この関数もシステムで移動を行った場合同様、移動後の座標がitemptrに返りますが、ルーベ情報構造体には影響しませんので、自分で書き換えてやる必要があります。

ID 0はルーベボックスですが、ここでの動作は座標系が異なるだけで、ウィンドウ外の動作と同じです。そこで、back_ground()関数の宣言を変更して、すでに実座標に変換してある座標を渡すだけで済ませてしましましょう。とりあえずback_ground()関数は置いておいて、次はID 1~4のスクロールボタンです。これらの処理はもう特に問題はないですね。ルーベ情報構造体の被拡大領域を移動させておいて、PAINTLUPE()で描画するだけです。その後のID 5~8の倍率ボタンですが、同じような処理ですし、関数内からでも引数1の下位8ビットを見ることで、どのボタンが押されたかを判断できますので、4つともひとつの関数button_05()にまとめてしましましょう。ここでも特に問題はありませぬ。倍率ボタンの反転を行ってから、ルーベ情報構造体の被拡大率領域を左上固定で変更し、ルーベを再描画しています。

ID 9はカレントカラーを表示するためのものですので、クリックされた場合の処理は無視して、最後はウィンドウ外の処理です。先ほどback_ground()関数の宣言を変更しましたので、ID 11で記述されている関数でも座標を渡すようにしてやります。

関数内では、まず前回と同じ座標ならば弾くようにして

あります。これはずっとボタンを押しっぱなしにしていた場合の対処です。一見、同じ色で同じところを何度処理しても同じように思えますが、アナログマスクがあるところではどんどん濃くなってしまからです。

ただし、これでは同じところを何度もクリックした場合も弾かれてしまいますので、main()関数でマウスが押されていなかった場合に、前回の座標を保持する変数bxとbyをリセットしています。そのあとは、左ボタンが押された場合と右ボタンが押された場合では、処理内容が異なりますので、場合分けをしています。左ボタンの場合ではカレントカラーで描画ですが、作業画面にマスクがあった場合には、前項で説明したPointSetWithMask0()を使用することになります。

マスクがなかった場合は、待避画面にはカレントカラーデータを単純に代入するだけでよいのですが、ルーベ内と表示画面にも描画する必要があります。ルーベ内の描画には、PAINTLUPE()では遅くなってしまいますので、EXコール\$2BのLUPESET()を使用します。この関数は、指定された実座標に対応するルーベ内のピクセルを指定されたカラーで描画しますが、実座標がルーベ内にないときにはなにも行いません。

あとは表示画面ですが、ウィンドウの上になで書き込んでしまつては困りますので、ウィンドウ座標を調べて避けるようにしましょう。同じプログラム内ならばltemptrからウィンドウ座標を得ることができですが、ここではEXコール\$25のGETEXISTWINDOW()で現在のウィンドウ情報を調べてみましょう。この関数は戻り値としてウィンドウ情報構造体のアドレスを返し、ウィンドウ情報構造体は以下のメンバで構成されています。

```
sExistWindow {
    int    x1;      ウィンドウ左上X座標
    int    y1;      ウィンドウ左上Y座標
    int    x2;      ウィンドウ右下X座標
    int    y2;      ウィンドウ右下Y座標
    sExistWindow *next;  次のウィンドウ情報
                        構造体へのポインタ
}
```

EX-Systemはシングルウィンドウですので、nextというのは無視しても構いません。それ以外の4つのメンバと照らし合わせて、座標がウィンドウの外であれば表示画面にも描画するようにしましょう。

右ボタンが押された場合は、その座標のカラーをカレントカラーとします。カラーは待避画面から最下位ビットをクリアし、そのカラーでカレントカラーボックスを塗り潰してやりましょう。

これで完成です。コンパイル時には、About.c同様にdeplib.oとDraw.hが必要ですが、それ以外にRevBox.sも必要になります。実行してみて、ルーベがスクロールするか、ルーベ内に描画を行った場合に表示画面にも描画されるか、またはその反対も描画されるかなどを確かめてください。

今回のプログラムでは、スケルトンとの対比を取りやすいように、呼ばれていない関数や意味をなさないcaseなどをそのままにしておきましたが、これはもちろん削除しても構いません。また、被拡大領域ボックスを直接ドラッグできないとか、連続した曲線を描けないなどの手抜きがありました。それらは各自で改良してみてください。

4. 5. 3. テキストビュー (TextViewer.c)

ファイラーからテキストファイルを選択し、内容を表示します。ここではプログラム内でウィンドウの作成などを行わずに、EXコール\$03のFILEWIN()を使用することでファイラーを呼び出します。また、テキストファイルの内容を表示するのに、システムがコンソールログを表示するのに使用しているEXType.xを呼び出すことにします。

FILEWIN()は、2種類の拡張子のファイルを切り換えて扱うことができ、単独でファイラーの表示から、ファイルの選択、ロード/セーブの指定までを行い、呼び出す側から見ればウィンドウを意識する必要はありません。引数はファイルウィンドウ構造体へのアドレスで、以下のようなメンバで構成されています。

```
struct WINPTR {
    char *title;      タイトル名へのポインタ
    char *ftype1;     ファイルタイプ名1へのポインタ
    char *fname1;     表示ファイル名1へのポインタ
    char *ftype2;     ファイルタイプ名2へのポインタ
    char *fname2;     表示ファイル名2へのポインタ
    char *fname;      選択格納領域へのポインタ
}
```

タイトル名というのは、ウィンドウのタイトルバーに表示される文字列ですが、“タイトル名.EX”というレジュームファイルを作成しますので、スペースなどのファイル名として使用できない文字は含めないでください。

ファイルタイプ名とは、ファイルタイプを切り換えるボタンに表示する文字列で、ファイルタイプが1種類しか扱う必要のない場合は、ファイルタイプ名2にヌルを入れます。具体的には、

```
winptr.ftype2 = NULL;
```

ではなく、

```
winptr.ftype2 = "¥0";
```

のように、ちょっと変な仕様になっています。

表示ファイル名は、ワイルドカードを使ったファイル名で、必ず“*.拡張子(半角3文字以内)”，もしくは拡張子の指定が必要なければ“*.*”を指定してください。FILEWIN()を終了した時点で、ファイラーで選択されたファイル名はfnameで示すアドレスに格納されていますが、これはシステム側で確保された領域を指していますので、あらかじめ領域を確保しておく必要はありませんが、次にFILEWIN()を読んだ時点で前のファイル名は破棄されてしまうという、これもちょっと変な仕様になっています。

また、戻り値は以下のようにになっています。

```
-1...ESCキーで終了
0...クローズボックスで終了
1...ファイルタイプ1でロード
2...ファイルタイプ1でセーブ
3...ファイルタイプ2でロード
4...ファイルタイプ2でセーブ
```

さて、ここではテキストファイルを扱いたいので、拡張子は“DOC”と“TXT”の2つを扱うことにしましょう。winptrに文字列を設定して、FILEWIN()を呼べば、あとは戻り値を調べるだけで、どのファイルのロード/セーブを指定されたかがわかります。今回はビューですので、セーブを指定された場合はメッセージを出してセーブはできないことを知らせましょう。ロードの場合には、EXType.

xに表示したいファイル名を渡して実行します。EXType.
xは、ESCキーで終了した場合は65535を、通常終了した
場合は2を返しますので、それを考慮してやれば、もうで
きあがりです。驚異の他力本願プログラムですね。

実際に実行したものが、図4.5のファイラーと、図4.6の
テキスト表示です。

4. 6. システム描画ルーチンの利用

最後は、システム側の描画ルーチン呼び出して、作業
画面に描画を行ってみましょう。

4. 6. 1. カレントドロー (CurDraw.c)

カレントドローモードで描画を行います。また、画面隅
に邪魔にならないようにマウスの座標を表示することにし
ましょう。描画を行うにはEXコール\$26のDRAW()を使
用しますが、その前にEXコール\$33のLOADDRAW()でド
ロールーチンをメモリ上にロードしなければなりません。
LOADDRAW()の引数にはドローモードを指定しますが、
ここではカレントドローモードで描画を行いますので、E
Xコール\$24のGETDRAWMODE()を使用してカレントド
ローモードを得ます。ここで、もし特定のドローモードで
描画を行いたいのであれば、そのドローモードをLOADD
RAW()の引数としてやることで、それがカレントドロー
モードになり、ロードされます。

描画の開始については、DRAW()を呼ぶだけで構いま
せん。仮にカレントドローモードが直線だったとしても、
DRAW()には始点だけを渡せば、終点の指定と描画は勝
手にやってくれます。ほかのドローモードでも最初の点を
渡すだけで、その先の指定や各画面の更新、ルーベがあれば
その中の描画まで自動ですので、プログラムではなにも
意識する必要はありません。また、DRAW()の第3引数
には、DRAW()内から逐一ジャンプさせるアドレスを指
定します。特に必要なければNULLを指定しておけばいい
のですが、ここではマウス座標の表示をさせることにしま
す。

マウス座標を表示させる関数がDigit()です。この関数
内では、マウスの位置によって座標を画面4隅のどこかに
表示させているのですが、DRAW()内からも呼ばれます
ので、そちらの仕様にも合わせておく必要があります。こ
こでは使用していませんが、DRAW()から呼ばれる関数
はint型の引数をひとつ取ることができ、指定段階を示し
ています。

たとえば、リサイズコピーの場合、転送元の1点目は
DRAW()が呼ばれたときにすでに指定は済んでいますの
で、転送元の2点目の指定中は1、転送先の1点目なら2、
転送先の2点目は3といった具合です。

また、戻り値は通常は0ですが、なにかの都合で画面全
体が書き換わり、ボックス指定中に表示される反転色の矩
形などが消されてしまった場合には、1を返すようにしま
す。今回は無条件で0でいいでしょう。

ClrDig()関数は表示した座標を消す関数です。BUF2G
RAM()を使って座標が表示されている領域を待避画面で
塗り潰しています。この方法では、座標の表示されている
領域を反転矩形や反転直線などが横切ったときにごみが残
ってしまうことがあるのですが、描画は影響を受けないの

で、今回は無視することになりました。ただ、描画が終わっ
ても残っていると問題なので、DRAW()関数から帰った
ら再び待避画面で塗り潰して、消すようにしてあります。

ルーベは使っていませんが、座標を拾ったあとにLUPE
2SCREEN()を呼んでいます。こうしておけば、仮にルー
ベを使っている外部ファイルが、ルーベを開いたままこの
プログラムを呼び出した場合にも対処できます。実際には、
その場合は今度は座標の表示が問題になってきますが。

5. とにかく使ってみよう

中野修一

EX-Systemはさまざまなパーツの集合体です。パーツごとに使用の選択ができますから非常に拡張性が高くなっています。半面、各パーツ間の行き来はやや煩雑で、インタフェイスの統一も取りにくいという欠点も持っています。現時点でサポートされている機能というのはたくさんありますが、全体像をつかむのが難しく、どのツールがなにをするためのものなのかわかりにくいという状況になっています。

これらを束ねるユーザーインタフェイスは独特なものになっています。実際に装備されている機能は豊富なのですが、とにかく慣れるまで、なにがどこにあるのかわからないでしょうし、特殊な用途の機能もありますので、どういう局面で使用するのものなのかというのは身体で覚えるしかありません。

このようなシステムでもいくつかの原則があります。

鉄則は、

「裏画面は通常は破壊されない」

ということです。どうしても必要な場合にはその旨のメッセージが表示されます。

たとえば画像の合成では、表と裏に置くべき画像はどうなるのかというのがわかりにくくなっていますが、これを理解していれば、事前に加工が入る場合には加工されるものを表画面に置かなければならないということがわかるでしょう。

操作系については、クリックは右クリックでデクリメント、左クリックでインクリメントというのが基本になっています。ごく一部に例外もありますが（画面表示との整合をとため）、基本はこうです。ドライブ番号などに対しても有効ですので覚えておいてください。そのほかの基本操作としてのCTRLキーやXF1キー、スペースキーなどはとにかく覚えてもらうしかありません。

とにかく使ってみてください。

5. 1. ペンを使う

まず、もっとも基本になるペン描画を見てみましょう。

●ペン濃度の指定

ペン描画は登録されているペンパターンを使ってフリーハンドで描画することや、ライン、ボックス、サークルなどが指定できます。

ペン描画では登録してあるパターンの濃度を変えて使うことが可能です。濃くすることはできませんので、あらかじめ濃いめのパターンを揃えておくほうがよいでしょう。

内部処理としては、単にビットシフトしているだけなので、色の変化を正確につかむことが困難だという問題もあります。薄めの色を塗る場合は（色にもよりますが）濃度3くらいが適当でしょう。

●UNDOペン

たとえば、ペンでの描画が主線をはみ出したりしてしまうことがあります。そういったときにはCTRLキーを押しながらペン操作をすることでUNDO画面から色を拾って修正することが可能です。

UNDOペンにもペン濃度が影響します。薄めの色で描画中に薄めの色でUNDOすることで自在な階調変化をつけることができます。

●裏画面カラー

エフェクトをかけた画像をUNDOペンで修正するときには、UNDOペンを気をつけて使わねばなりません。ツールを変えた途端に、ついFIXされてしまうことがあるからです。このような事態を避けるにはエフェクトをかけるまえの画像を保存しておいて、その画像を裏画面に置いておき、裏画面カラーを指定してこれをUNDOパッファの代わりにしてやります。なお、この場合はCTRL併用のUNDOペンがエフェクトを強くする意味になります。

●ドット修正

広い範囲のドット修正ではカスタムウィンドウのルーベモードが使いやすいでしょう。通常ルーベにも全画面モードというのがありますが、機能の切り換えなどではカスタムルーベのほうが有利になります。

カスタムルーベの利点のひとつはグラデーションパレットが使えることです。

グラデーションパレットは4段階グラデーションの色を決めるものです。その一部を切り出してパレットとして使用しています。これを使えば中間色を簡単に作成することができます（というより、これがないと中間色が非常に使いにくかった）。手作業でのアンチエイリアスではこういったものが必須となるでしょう。

カスタムルーベモードではグラデーションなど、一部の処理がリアルタイムに画面に反映されなくなります。内部処理を終了してから一気に画面に書き出すことになります。

仕上がりを確認しながら作業するには通常のルーベを使用してください。ルーベはウィンドウモードと全画面モードに簡単に切り換えることができます。このへんは好みと慣れの問題ですね。

また、カスタムルーベ状態では、エフェクトをかけることは基本的にできません。そういった微妙な部分の用途にはスーパールーベを使用してください。

5. 2. 合成の基本

●もっとも簡単な合成

なにかキャラクターなどの絵を描き、背景と合成します。Z's-EX時代からご存じの方にはお馴染みですが、そうでない人にはわかりにくいPERSPECTIVEというのを使います。

もともとPERSPECTIVEは裏画面にある絵を3次的に回転して張り付けるという機能ですが、まったく回転してない状態で張り付けると単純合成になることから、EX-Systemでは単純合成は伝統的にこれを使用しています。

キャラクターの背景部分にマスクをかけることを忘れなければ単純な作業です。

マスクは表画面にも裏画面にも指定することができます。

●アンチエイリアス

単純に背景にベタマスクを敷いて合成するとギザギザが目立って綺麗ではありません。できるだけ綺麗に合成することを考えてみましょう。

まず合成する画像は大きめに作成しておきます。周りにマスクをかけ、マスクごと縮小してやれば縁のギザギザしない合成ができます。このとき、ツールメニューで編集対

象にCOLORとMASKの両方を設定することを忘れないでください。

これはパターンブラシやスタンプパターンの登録時にも有効な技となります。

ラインのはっきりしているものや複製するものが大きすぎて縮小できないときは、合成してから輪郭を修正してやります。いくつか方法がありますので、場合に応じて使い分けてください。

まず、PPeDitを使用する場合です。

適当な大きさのペン先を選び、SHADEを指定してなぞってやれば継ぎ目が綺麗につながるはず。ぼかし部分のはみ出した場合は右クリックで修正できます。

次に裏画面との合成を行う方法です。

まず、画像を裏画面にコピーし、全体にぼかしをかけます。ぼかしのかかったものを裏画面に送り、パレットコントロールで裏画面カラーを選択したうえで、輪郭部分をなぞっていきます。これでなぞった部分だけ裏画面の画像が表に出てくることになります。この場合はCTRLキーを押すとUNDOペンの動作になります。

●さらに賢い使い方

ソフトフォーカスなどのエフェクトなどをかけたときの処理法です。まず、ツールをペンに変え、パレットで裏画面カラーを選択して、裏画面に元画像を置いておきます。これで準備は完了。

エフェクトを強めにかけます。登録キーでしっかりFIXしてください。

ペンツールを呼び出して、エフェクトをかけすぎた部分を裏画面カラーで削ります。このとき、薄いペンなら軽く削ることができます。削りすぎたときはCTRLキーを押してUNDO画面から色を戻します。このときもペンの濃さは有効になります。

このようにすればペンでタッチをつけながらエフェクトのかかり具合を自在にコントロールすることができます。

●スキャナからの取り込み画像を彩色する

直接合成とは関係ありませんが、スキャナで取り込んだ輪郭のはっきりしていない画像を塗っていくときにはいくつかの方法論が考えられます。以下に例を挙げますので、場合に応じて使い分けてください。

1) 普通にペイントして修正する

作業は単純です。塗り残した部分はルーベで潰してまわります。アンチエイリアスをかけるのが非常に面倒です。

2) LineReviseを使う

取り込み線を2値化して、普通のペイントで塗り、輪郭線をアンチエイリアスしてやる方法です。取り込み画像に限らず使われる方法ですね。この場合は多少の誤動作を覚悟し、ペンタッチを捨てることになります。

おおむね綺麗な出力にはなりますので、UNDOペンなどで細部を修正して仕上げていってください。

3) MagicWandを使う

塗りたい部分をMagicWandで囲み、内部をマスクペイントしたうえでマスクを反転します。あとは好きに塗るだけ……。

手順は多くなりますが、細部の塗り分けにグラデを使ったり、ペンタッチなどを多く加えたいときには非常に有効です。ただし、アンチエイリアスが完璧に行われるとは限りません。範囲の内部に描かれたものは、丁寧にマスクするか、あとから合成し直すなどの必要があります。

4) 主線合成を使う

取り込んだ絵の上から境界線を潰して塗り込んでいき、塗り終わったら上から境界線を合成し直してやるということです。ペンタッチをそのまま生かすにはよい方法です。

薄い線で書かれていた場合、塗り絵した側の主線をあらかじめ消しておかないと、合成後にその部分の線が濃くなってしまいます。

5) 線画マスク変換を使う

3)のバリエーションです。取り込んだ絵に線画マスク変換をかけると主線の部分がいちばん薄いマスクに変換されます。その上からどんどん色を塗っていき、最後に主線合成をして仕上げます。

いちばん薄いマスクは描画にはほとんど影響しませんので、それでアタリだけとって塗り込んでいくことができます。描画作業時にXF1キーを押すとマスクを一時的に透かして見るすることができます。

広い範囲に取り込み画が塗り込んであると、マスクが邪魔で作業が難しくなります。下絵のほうで調整してください。また、線が薄すぎて輪郭がうまく取れないときは明度シフトを行ったうえで実行してみてください。

6) 細線化を使う

取り込み画像を細線化し、それぞれの領域を塗っていきます。だいたい塗り終えたら主線合成でもとの線を合成していきます。普通のペイントだけで塗っていけるのでわかりやすく、作業量自体も少なくてすみます。

取り込み画像が濃くはっきりした線でない、合成したときに細線化された線が浮いて見えることがあります。これを補正するには境界線を残さないように塗ったものと主線を合成する必要があります。また黒く塗りつぶされた領域はうまく変換されないで、下絵を描くときにそれを考慮しておく必要があります。

7) スーパールーベ/仮想画面を使う

2)のデジタルで塗り分ける処理と似ています。4倍の大きさの絵を描き、縮小してアンチエイリアスをかけていくというものです。かなり綺麗には仕上がりますが、大きな領域の描画や効果などが不可能で、作業量も多くなります。

●いろいろな合成

・質感合成

テクスチャ画像として指定したものの明暗を適当な帯域の明度差として画像に加えていきます。

処理前にテクスチャーに加工が施されますので画面が破壊されることになります。テクスチャーを表画面に、画像を裏画面に置くことを忘れないようにしてください（まあ、逆でもそれなりの効果は出ますが）。

・プレーン分離/合成

彩度を上げたり、輝度を調整するといった当たり前の使い方のほかにも、RGBプレーン、HSVプレーンごとに分解された映像をエフェクトをかけたあとで再合成することで、通常のエフェクタだけでは不可能な処理が実現できます。

RGBやHSV成分に分離した絵を元のプレーンではなく別のプレーンとして合成することもできますし、別の絵の成分を重ねあわせるといったことも面白い効果となりますでしょう。

・主線合成

すでに説明しましたが、スキャナで取り込んだ線画を合成するための専用命令もあります。単に表画面と裏画面の掛け算を行うだけのものですが、こういった目的専用のものですから塗り込んだ絵に手描きの線を生かすことができます。

あくまでもスキャナで取り込まれた白地に黒で描かれた絵を前提として処理していますので、ほかの用途に応用して使うことは困難かもしれません。

5. 3. 雲を描く

手描きで自然物を描くというのはなかなか大変なものです。作為的でない形状にはフラクタルやランダムな要素を含んだツールを使用すると効果が上がります。EX-Systemには通常のペンなどのほかに、そういったものを描くためのツールがいくつか用意されています。

●基本方針

自然物はフラクタル系のツールを使って作成するのが簡単でよいでしょう。グラデーションにフラクタルかをかけるだけでも「やもやしたもの」ができることは有名ですが、そういったものを局所的に制御することでさらにメリハリのあるものが作れます。フラクタルの使い方ですが、基本的に縦方向にしかかかりませんので、重ねてかけたりすると方向性が過度に表れすぎてしまうことがあります。画像を横倒ししてフラクタルをかけることは基本的なテクニックのひとつです。

また、ランダムフラクタルのもとになるRF.DATはRF.BUILD.Xで作成することができます。場合によっては強度の違うRF.DATをうまく使い分けることがポイントになることもあるでしょう。とりあえず「非常に弱いRF.DAT」を作っておくことをおすすめします。

絹雲

ペンやパターンブラシで基本的な形状を置き、モーションプレー系のフィルタで処理します。平行系または回転系を用います。

積雲

パターンブラシで作るのがもっとも簡単だと思われます。デフォルトで雲っぽいものはいくつか用意してありますし、エディットすればもっと多彩なものを揃えることも可能です。上部はもこもこで、下部は平らという基本形状を押さえていれば、なんとかそれらしくなるでしょう。初めに影になる部分の色で描き始め、あとから普通の部分やハイライトを乗せていきます。

またはエアブラシ（フラクタルつき）を使い、色を重ねていきます。写真などを見ながら大まかな感覚をつかんでいくことが重要です。

ちなみに川原氏はタブレットと普通のペンでひたすら手描きしているようですが……。

5. 4. 自分のパターンブラシを作る

たぶん慣れないとアナログマスクの動作がわかりにくいと思います。アナログマスクを使ったパターンブラシを作成するための手順を挙げておきましょう。

1) マスクパターンを作る

まず白や灰色系の色で濃淡をつけてパターンを作成します。コツは薄めに描くこと、大きめに描くことです。パタ

ーンブラシで取り込めるサイズは64×64ドットまでですが、この大きさにパターンを作るのは困難なことがあります。あとでリサイズするほうがよいでしょう。

2) マスクに変換する

できあがったらマスクの項目から「緑→マスク」を選択します。裏画面に移って白で初期化し、XF4でマスクを取り込んで、XF3でマスクを反転しておきます。

3) ブラシを取り込む

必要に応じてリサイズなどを行い、パターンブラシのGETで取り込んでいきます。GETボタンを押し、いちばん大きい範囲を指定してください。アイコンが取り込まれますので、いま取り込んだアイコンをダブルクリックします。別ウィンドウが開いてパターンが表示されます。この状態で画面をクリックすると追加のパターンを取り込むことができます。間違って取り込んだ場合はそのパターンをダブルクリックすることで削除できます。

6. EX用ウィンドウデザイナー

吉田 泉

EX-Systemにはウィンドウを操作するEXコールも用意されていますが、ウィンドウそのものを手作業で作るのは大変手間のかかることです。また、ウィンドウを扱う部分のコーディングはどんな外部ファイルでもほとんど変わりませんので、毎回同じようなことを書くのは面倒です。そこで、ウィンドウデザイナーの登場です。このウィンドウデザイナーを使えば、ウィンドウのデザインをマウスを使って手軽にかつ視覚的に行うことができ、デザインしたウィンドウからC言語の外部ファイルスケルトンを生成することができます。

デザイナーの起動に特殊な環境は不要です（生成されたソースをコンパイルするためには、第4章で示された環境が必要です）。本体であるEX_DES.Xを起動するだけです。起動すると、図6.1のような画面が表示されます。画面左には、タイトルバーとクローズボックスだけのウィンドウがあり、画面右のコマンドを選択して、ここにマウスでいろいろなコントロールアイテムを配置していくことになります。

6.1 コマンド

●ITEM

ウィンドウ上に配置する（された）コントロールアイテムを操作するコマンドです。

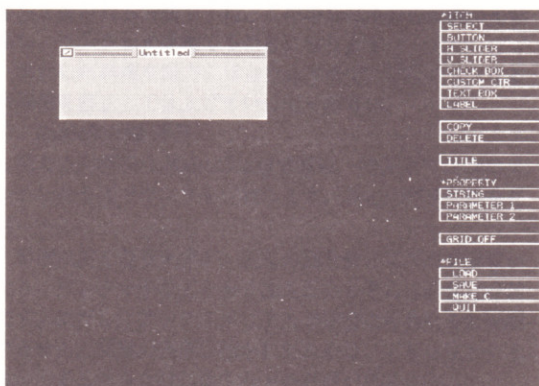
・SELECT

ウィンドウ上に配置されたコントロールアイテムを選択します。選択されたコントロールは、枠が反転表示されます。このときに枠の四隅をドラッグすることにより、各コントロールのリサイズを、枠の中をドラッグすることでコントロールの移動を行うことができます（チェックボックスはリサイズできません）。

・BUTTON~LABEL

それぞれのコントロールアイテムウィンドウに配置します。このコマンドを選択したあとに、ウィンドウ上のアイテムを置きたい場所でマウスの左ボタンを押すと、枠が表示されますので、そのままドラッグしてアイテムの大きさを決定してください。コントロールアイテムについては後述します。

図6.1 デザイナーの起動画面



・COPY

選択されているコントロールをコピーします。コピーされたコントロールは、元のコントロールと同じ位置に重なって配置されますので、適当な位置に移動させてください。

・DELETE

選択されているコントロールを削除します。

・TITLE

ウィンドウのタイトル文字列を変更します。このコマンドを選択すると、画面に文字入力を促すプロンプトが表示されますので、63バイト以内で入力してください。

●PROPERTY

コントロールのプロパティを決定します。選択されているコントロールに文字列（STRING/35バイト以内）と2つの数値（PARAMETER 1・2）を設定することができます。それぞれの意味については後述します。

・GRID

コントロール配置時のグリッドを、0～9の値で設定します。0を設定した場合は、OFFになります。

●FILE

ファイル操作関係のコマンドです。

・LOAD

以前にセーブされたウィンドウデータをロードします。

・SAVE

デザインされたウィンドウデータをファイルにセーブします。ここでセーブしたデータは、LOADで読み込むことができます。

・MAKE C

デザインされたウィンドウデータを、C言語のスケルトンに変換して、ファイルに出力します。ここで出力されたファイルは、LOADで読み込むことはできません。

・QUIT

デザイナーを終了します。

特殊な操作としては、ウィンドウの右下隅をドラッグすることでウィンドウのサイズを変えることができます。また、そのときにアクティブな情報が画面右下に表示されますので参考にするといいいでしょう。

6.2. コントロールアイテム

それぞれのコントロールアイテムの詳細と、それぞれのプロパティについて説明します（表記のないプロパティは特に意味を持ちません）。また、コントロールの総数は64個までです。

●BUTTON

コマンドボタンです。実行時にこのボタンがマウスによって押されると、特定の関数が呼び出されます。また、マウスがボタンの上にくるとボタンが反転します。反転させたくない場合は、生成されたソースのitem[]配列中の値を変更します。詳しくは「4.5. ウィンドウを開く外部ファイルの作成」を参照してください。

・STRING

ボタンの中に表示される文字列を設定します。

・PARAMETER1

文字列の表示法を指定します。0で左寄せ、1で中央、2で右寄せになります。

・PARAMETER2

文字の色をカラーコードで指定します。

●H/V SLIDER

水平、あるいは垂直のスライダです。実行時には、スライダの中をドラッグすることで、スライダの値を変更することができます。

・PARAMETER1

絶対値でスライダの色を、符号で左右または上下のどちらをスライダの値とするかを設定します。

・PARAMETER2

スライダの初期値を設定します。スライダの値は、0からそのスライダの長さ（ドット）までです。ただし、デザイン上では反映されません。

●CHECK BOX

チェックボックスです。実行時には、マウスのクリックで、ON/OFFがトグルで切り換わります。

・PARAMETER1

チェックボックスの初期値です。0でOFF, 1でONです。

●CUSTOM CTR

自由に使えるコントロールですが、内容については自前でやらなくてはなりません。

・PARAMETER1

0以外のときには、枠を表示します。

・PARAMETER2

0以外の時には、マウスがコントロールの上に来たときに反転します。

●TEXT BOX

文字列入力のコントロールです。実行時には、キーボードから任意の文字を入力できます。コマンドボタン同様、マウスがテキストボックスの上にくると反転します。

・STRING

文字列の初期値を指定します。ただし、デザイン上では反映されません。

●LABEL

ウィンドウ上に文字を表示します。表示するだけです。で、ラベルをクリックした場合になんらかの処理を行いたい場合は、生成されたソースに手を加える必要があります。

・STRING

表示される文字列です。

・PARAMETER1

絶対値で0のときは右寄せ、1で中央、2で左寄せで、負数の場合は影が付きま。

・PARAMETER2

文字列の色を指定します。

カレントからインクルードするように記述されています）。

具体的に、サンプルのスケルトンに肉付けをしなが説明していきましょう。サンプルとして、図6.2のようなウィンドウを用意しました。ディスク中にSample.resというファイル名で保存してありますので、プロパティなどについてはデザイナーでロードして確認してください。

このサンプルは、ウィンドウ上で設定したカラーで任意のカラー境界あるいはマスク境界の領域をペイントするというものです。ここで、それぞれのコントロールの挙動をはっきり定義しておきましょう。まず、3本のスライダですが、これはそれぞれペイントするカラーのRGBの値を示します。その右のテキストボックスはRGBを数値で表します。RGBそれぞれのスライダとテキストボックスはリンクしていて、どちらか片方を操作した場合は、もう片方もそれに合わせて変化しなければなりません。

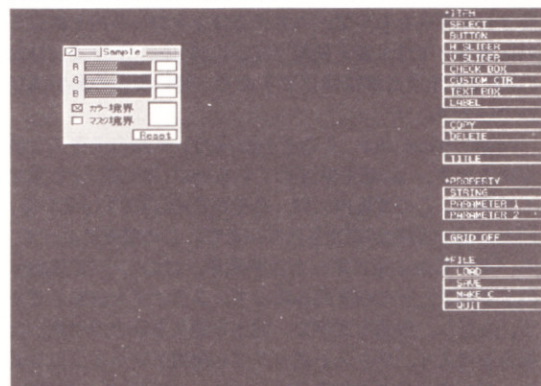
また、左の'R','G','B'のラベルをクリックした場合も、値を変化するようにしてみましょう。たとえば、右クリックで増加、左クリックで減少といった具合です。もちろん、それに合わせてスライダとテキストボックスも変化します。これらのコントロールで設定したカラーは、その右下のカスタムコントロール内に表示し、確認できるようにします。ついでに、このカスタムコントロールをクリックした場合はRGBすべてを増減するようでもしましょう。左下のチェックボックスは境界を選択するのですが、どちらかしか選択できないラジオボタン式にしてみます。また、使いやすさを考慮して、ラベルをクリックしてもチェックボックスに影響を与えるようにします。右下のRESETボタンは、特に必要性があるとも思えませんが、すべての設定をリセットします。

このサンプルをCに変換し、肉付けを行ったのが、Sample.cです。とりあえず、ソースの各部位について簡単に説明を入れます。まず、17行目から29行目までが、それぞれのコントロールが選択されたときに呼び出される関数のプロトタイプ宣言です。それぞれ”コントロール名+アンダースコア+コントロールごとの2桁の通し数字”で表されていますが、デザイナー自身はラベルには関数を振り当てないので、ラベルになにか機能を持たせたい場合は、自分で関数を定義してやらなければなりません。いまの場合では、'R','G','B'のラベルに、label_01~03という関数を付け加えてやります。ただし、チェックボックスに付随するラベルに対応する関数は定義しなくて構いません。これについてはあとで説明することになります。30行目は、ウ

6. 3. C言語への変換

ウィンドウのデザインがひととおり終了したら、[MAKE]コマンドでC言語のスケルトンを生成します。ただし、終了する前にあとで修正することを考えてセーブしておいたほうがよいでしょう。生成されたソースは、そのままコンパイルして、EX-System上で実行することができます。一度実行させてみて、動作を確認しておくのもいいでしょう。スケルトンのままでも、スライダやチェックボックス、テキストボックスは動作します。これは、スケルトン内でそれらの動作を行う関数と呼んでいるからなのですが、それらの関数はdeslib.cに記述されていますので、コンパイル時には必ずdeslib.cをリンクしてください（関数はdeslib.hで定義されていますが、スケルトンはこれを

図6.2 サンプルプログラム



ウィンドウの外をクリックした場合などに呼び出される関数です。

34行目からのITEM型配列item[]は、それぞれのアイテムの座標等を、56行目からの構造体ctrl[]はコントロールのプロパティを示しています。ここで注意が必要なのは、コントロールIDは、アイテム番号よりも2小さいということです。これは、クローズボックスとタイトルバーもアイテムに含まれているからです。ITEM型の内容については、「4. 5. ウィンドウを開く外部ファイルの作成」で説明されていますので、ここでは説明しません。ctrl[]構造体は、デザイナ上でのプロパティに相当します。各プロパティは(nはコントロールID)、カスタムコントロールのPARAMETER2を除いて、

```
STRING      → ctrl[n].str  
PARAMETER1  → ctrl[n].para[0]  
PARAMETER2  → ctrl[n].para[1]
```

というメンバに対応しています。

メイン関数では、まずアイテムを設定して、ウィンドウを開き、各コントロールを表示する作業を行っています。ここで、set_window_item()とshow_all_ctrl()関数はdeslib.cで記述されている関数で、それぞれデザイナ関数側にアイテム情報を伝える、コントロールを表示するという機能を担っています。

94行目からのforループが本処理です。EXコールのMANAGE()でイベントを拾い、イベントの起きたアイテムをswitchで選別し、それぞれの関数に飛ばしています。このswitch文の中もほとんど自動で生成してくれますが、先程も言ったようにラベルに機能を持たせたい場合は自分で書く必要があります。ここでは、case2~case4の処理がそれに当たります。また、case5~case7のスライダの処理内には、h_slider()というデザイナ関数があらかじめ記述されています。この関数は、マウスの座標からスライダの値を決定し、para[1]に格納してくれますので、h_slider_01~03は、あらかじめ値がpara[1]に格納されて呼ばれることになります。したがって、h_slider_01~03内では、para[0]から値を読み出し、それをテキストボックスとカスタムコントロールに表示するだけでいいわけです。

h_slider_01()関数を見てみましょう。元々はこの関数は空の関数だったのですが、非常に簡単に済んでしまっています。RGBの各レベルは0~31までですが、スライダは倍の長さに取りましたので、値を拾うときにはpara[1]を半分にします。その値をテキストボックスのstrメンバに転送し、show_ctrl()というデザイナの関数で表示します。最後のfill_color()というのは、新たにユーザー関数として作ったもので、カスタムコントロール内を設定されたカラーで塗り潰す関数です。h_slider_02(), h_slider_03()も処理内容は同じですので、特に説明の必要はないでしょう。

case8~case10は、それぞれRGBのレベルのテキストボックスです。case8で呼ばれているtext_box_01()を見て下さい。この関数内には、あらかじめLEDIT()という、EXコールの関数が記述されています。この関数によって、キーボードから受け取った文字列が、strメンバに格納されますので、その値を読み取って、スライダを移動させます。ただし、無効な文字や、範囲を越える値を受け取った場合を考えて、テキストボックスも再表示するようにしましょう。スライダを移動させるには、para[1]メンバにス

ライダの値、この場合はRレベルの2倍の値を代入し、show_ctrl()関数で表示します。この関数は、テキストボックスの表示にも使いましたが、コントロールの種類に限らず、引数で指定されたコントロールIDに対応するコントロールを表示するものです。したがって、コントロールのプロパティを変更するには、基本的に変更したメンバを書き換え、show_ctrl()を呼べばいいことになります。ここではやりませんが、ラベルやボタンの文字も書き換えることができます。

さてここで、無視していたcase2~case4のラベルの処理を説明します。関数はリストの後ろのほうに付け加えたので、label_01()を見て下さい。ここまで説明しませんでしたでしたが、コントロールで振り分けられた関数は、引数にEXコール関数MANAGE()の戻り値の下位8ビットが渡されます。ここにはマウスのボタンの情報が格納されていますので、そこからマウスのどちらのボタンがクリックされたかを拾うことができます。詳しくは「4.5. ウィンドウを開く外部ファイルの作成」で説明されています。ここでは、右か左かだけでいいので、それぞれ'&'演算子で調べています(MS_LEFT, MS_RIGHTはEXLIB.Hで定義されています)。あとは特に問題はありませんね。スライダ移動とテキストボックス表示は、いままでと同じです。

case11, case12は、境界を選択するチェックボックスです。ここには、スライダと同様に、あらかじめcheck_box()というデザイナ関数が記述されています。この関数は、指定されたコントロールIDに対応するチェックボックスをトグル操作でON/OFFする関数で、ON/OFFの値がpara[0]メンバに入って帰ってきますので、通常はユーザー関数内(ここではcheck_box_01(), check_box_02())ではなくにもする必要はありません。しかし今回は、チェックボックスをラジオボタンのように動作させますので、check_box()関数をコメントアウトして、自前でやることにします。とはいっても、check_box_01()関数を見ればわかるように、自分をチェックし、もう片方のチェックをクリアしているだけです。

また、switch文の中に、case11と同じ処理としてcase13が、case12と同じ処理としてcase14が追加されています。この2つはチェックボックスに付随するラベルですが、「ラベルをクリックした場合もチェックボックスを選択したことにする」ということでしたから、チェックボックスとまったく同じ処理でいいことがわかりでしょう。これで、最初のプロトタイプ宣言で関数を追加しなかった理由がわかりましたね。

case13はカラーを表示するカスタムコントロールです。クリックした場合は、RGBすべてを増減するのでしたね。ちょっと長くなってしまいましたが、custom_ctrl_01()関数内で行っていることは、RGBのラベルでやったことをまとめただけです。特に説明することもないでしょう。

case16はリセットボタンでしたね。特に説明は必要ないと思いますが、すべての初期化ということで、ここではコントロール表示にshow_all_ctrl()関数を使いました。

case17はコントロールではなく、「すべてのコントロールを除くウィンドウ内領域」を示しています。特に気にする必要はありません。

case18はウィンドウ外領域を示します。つまり背景部分ですね。back_ground()関数内では、まずカラーコードを拾い、チェックボックスを調べてカラー境界かマスク境界

かで関数を呼び分けています。これらのペイント関数はEXコールで用意されているものですが、ウィンドウには対応していないので、いったんEXコールCLOSEWIN()関数でウィンドウを閉じています。このあたりのEXコール関数は「4. 外部ファイルの作成」を参照してください。

最後に、新たに作ったユーザー関数fill_color()について説明しておきましょう。これは、現在設定されているカラーでカスタムコントロールをペイントする関数でした。カラーの拾い方についてはいいでしょう。ペイントについては、標準関数のfill()を使いますが、コントロール内部領域の座標が必要になってきます。ウィンドウの左上座標は、itemptr構造体のx0とy0メンバが保持しており、ウィンドウの左上座標からコントロールまでの相対座標はitem[]配列に格納されていますので、その和でコントロールの絶対座標を求めることができます。

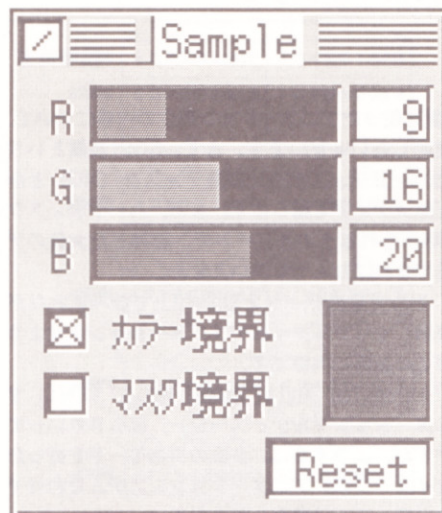
ただ、item[No.][2]～item[No.][5]には、コントロールの枠を含めた座標が入っていますので、これを使うと枠まで塗り潰してしまいます。実際にはこれよりも1ドット小さくしなければならぬのですが、マウスが上に来たときに反転するコントロールはitem[No.][6]～item[No.][9]に枠を含まないコントロールの座標が格納されています。いまの場合は、プロパティを変更して反転しないようになっていますが、カスタムコントロールにもこの値が格納されていますので、これを使うことにします（反転しないものは、この4値を使いませんので、自由な値を書き込んでおいても構いません）。そうとわかれれば簡単ですね。item[No.][6]～item[No.][9]には、順にコントロールの左上X座標、Y座標、右下X座標、Y座標が格納されていますので、itemptr.x0とitemptr.y0との和を取れば、コントロール内部の領域座標を得ることができます。

6. 4. 美しいユーザーインタフェイスを

以上でひとつのコントロールの説明が終わりました。しかし、このサンプルを実際に動かしてみると、いくつかの問題点が見つかります。まず、スライダ上でマウスボタンを押せばなしにすると、ドラッグしなくてもテキストボックスの値が書き変わり続け、点滅しているように見えてしまうこと。機能には問題ありませんが、あまり美しいものではありません。これを避けるには、RGBのレベルを保持する変数を新たに設け、スライダ関数が呼ばれた段階で、以前と同じレベルであれば弾くといったような工夫が必要になってきます。

また、RGBのラベルをクリックすると、あっというまに値が増減されていってしまいます。これでは、「1だけ上げよう」などといった場合に、非常に不便です。そこで、ラベルの関数内でウェイトを入れたり、あるいはitem[]配列内の復帰タイミングのフラグを立てておくなどの対処が考えられます。そのほかにも、スペースキーでの表裏画面の反転などを行いたい場合もあるでしょう（デザイナーが生成したスケルトンのデフォルトでは禁止されています）。これについては、今回はMANAGE()関数の引数を2にしてやるだけでなんの問題もありませんが、ウィンドウ外になんらかの目安を表示してある場合には、それなりの対処が必要になります。このように、必要な機能が満たされているからよしとせずに、細かい部分にも気を配って、美しいユーザーインタフェイスを心掛けるようにしましょう。

図6.3 サンプルの実行例



7, AMIシステム

福嶋章太

SCSI装置を使ったアニメーション再生の内容について、ちょっと説明したいと思います。まず、SCSI装置というのは、当然ハードディスクや光磁気ディスク（MO）とかの大容量記憶メディアを指します。まあ、ハードディスクやMOと同等かそれ以上のスピードで、連続した大量のデータを転送できればなんでも構いません。

そして、その大量のデータをX68000上で読み込みながら画面に表示してリアルタイムでアニメーションさせようというのがこの企画の内容です。

名前は「Animation, Multi Image」略して「AMI」です。「アニメーションマルチイメージ」、はっきりいってこじつけですが、ようするに、多数の再生モードを持った（マルチなイメージの）アニメーションシステムなわけです。文法的な突っ込みは勘弁してください。

当然ですが大容量のSCSI装置を前提としています。で、そのSCSI装置から、リアルタイムにデータを読み込みながらアニメーション再生をするわけです。

大容量のSCSI装置を前提としているだけあって、なにも考えず、どこどかと磁性面（？）を消費します。そうです、無圧縮です。私の作成したアニメーションファイルで、最大のが32Mバイトの大きさなのですが（これで1分程度の再生時間）、MOをフロッピーより狭く感じることができます（自慢にはならない？）。

無圧縮の利点もあります。まず、どんな複雑な画像データでも再生スピードの設定可能最大値（以下、最速値）には、影響が出ません。取り込みバリバリのアニメーションも、やりようによっては可能です。さらにX68000本体のスピードにも最速値は影響を受けません。10MHzの初代でも、ドーピング済みのX68030でも一緒です。さらにさらに、読み込んだデータはそのままVRAMに展開されるのでメモリをほとんど消費せずにアニメーションを再生できます。実行ファイルの収まる40Kバイト分も空いていれば十分なわけです。

そして、これがAMIシステムの最大の特徴であり最大の売りでもあるのですが、それは、AMIシステムの再生能力（最速値と最大再生時間）が、使用するSCSI装置に完全に依存するということです。要するに、使用するSCSI装置が速くなれば、その分AMIシステムの最速値が上がり、使用するSCSI装置が大容量になれば、その分AMIシステムの最大再生時間が長くなるわけです。

今後、SCSI装置はいま以上に高速化、大容量化が進むでしょうから、AMIシステムの再生能力は、ほっといても勝手に向上するわけです。らくちん、らくちん。

7. 1. SCSIとは

X68000のSCSIには外付けの拡張ボード（CZ-6BS1）によるものとSUPER以降の機種に標準で内蔵されたもの、満開製作所のSCSI2ボードの3種類があります。DOSコールやIOCSコールを使ってSCSI装置を扱うのであればこれらの違いをまったく気にする必要はありません。しかし、高速化などのためにIOCSコールなどを使わず、SPC

（SCSIプロトコルコントローラ）を直接操作してSCSI装置を扱おうとすると2種類のインタフェイスの違いをしっかりと理解しなくてはなりません。

具体的に違う点を挙げると、前者2つはSPCのポートやSCSI-ROMのアドレス、使用される割り込みのレベルとベクタ、SCSI-ROM識別用の文字列などになります。SPCのポートは外付けが\$EA0000～、内蔵が\$E96020～となっています。内部レジスタの配置は同じなので、ベースアドレスの違いにだけ注意しましょう。SCSI2ボードの場合はコントローラ自体が違います。

SCSI-ROMは外付けが\$EA0020～、内蔵が\$FC0000～となっています。中身はIPLが書かれている程度なので、今回は無視してもかまいません。割り込みレベルとベクタは、外付けはレベル2または4でベクタ\$F6、内蔵がレベル1でベクタ\$6Cとなっています。認識用文字列は、外付けは“SCSIEX”、内蔵は“SCSIIN”です。

今回のサンプルプログラムではIOCSコールを使ってSCSI装置を操作しているのでこれ以上は書きませんが、もっと知りたい人はソフトバンク発行の『InsideX68000』のSCSIの章に詳しく書かれているので、そちらを参照してください。

7. 2. SCSI用IOCSコール

SCSIをアセンブラレベルで簡単に操作するためのIOCSコールが、コール番号\$F5に用意されています。基本的にSCSIの操作はすべてこの\$F5で行います。細かい機能の選択は、SCSIコール番号というのが別に定義されていて、その番号をD1レジスタにセットするかたちになっています。以下にSCSI用IOCSコールを利用するプログラム例を示します。

```
moveq.l #00, d4
moveq.l #24, d1
moveq.l #f5, d0
trap #15
```

この場合、D4レジスタには操作対象のSCSI装置のID（0～7の範囲で指定）、D1にSCSIコール番号\$24（TESTUNIT:SCSI装置が動作可能であるか調べる）をセットしています。

SCSIコールは、低レベルコールと、高レベルコールとに分けることができます。低レベルコールは、かなりハードよりなルーチンで、SCSIのハード的な動作を理解していないと利用することができません。逆に、高レベルコールは、より一般的なコールで、READやWRITEといったコールひとつでデータの読み書きができるようなルーチンになっています。実際には、高レベルコールは内部で低レベルコールを使って処理を実行しているだけなので、高レベルコールはいくつかの低レベルコールを自動で呼び出してくれるコールと考えることができます。

SCSIコールの詳しい機能について知りたい人は、1992年のOh!X3月号にSCSIコールの機能表が載っているで、そちらを参照してください。

7. 3. アニメーションのデータ構造

AMIのアニメーションのデータ構造は、基本的にすべて無圧縮です。無圧縮ですから、ペタ塗りの動いているか

どうかかわらないようなアニメーションでも、自然画取り込みの激しく動きまわるアニメーションでも、ドット数や色数などの条件が同じなら、同じデータ量になります。

なぜ圧縮をしないかという、第1に、このアニメーションプログラムはSCSI装置から受け取ったデータをリアルタイムで画面に表示しなければならないので、圧縮データを展開するだけの時間的余裕がない、ということ、第2に、圧縮にはたいいて向き、不向きというものがある、不向きなデータは展開に時間がかかったり、圧縮したはずが逆に元データより容量が大きくなったりということが起こりうるという2つの理由で今回は無圧縮に決めました。

無圧縮ではありますが、このアニメーションのデータ構造は、多くの表示モードに対応できるかたちになっています。いまのところ、表示色は2色、4色、16色、256色、65536色、で表示サイズは64×64、128×128、256×256、512×512というモードをサポートしています。

では、実際どのようなデータの構造なのかというと、実は非常に単純な構造になっています。データはすべて512Kバイトを基本単位とします。すなわち、数コマのアニメーションデータをまとめて1データとして扱うことになります。ですから、256×256の2色データは64コマで1データ（1ドット1ビットのデータ）、128×128の65536色データは16コマで1データとなります。その1データを2色と4色はテキストVRAMに、16、256、65536色はグラフィックVRAMに、それぞれ図7.1のように並べます。

それぞれをVRAMアドレスの先頭からアドレス順に512Kバイト分眺めてみてください。ただし、グラフィックの16色と256色のモードは65536色モードのアドレス配置に直した状態で眺めます。すると、すでにそのデータの並びがこのアニメーションの1データ分のデータ構造になっているという仕掛けです。例として、256×256の256色のデータの並びを図7.2に示します。

どうです、かなり単純なデータ構造だということがわかってもらえたでしょうか。

図7.1 AMIデータの配列

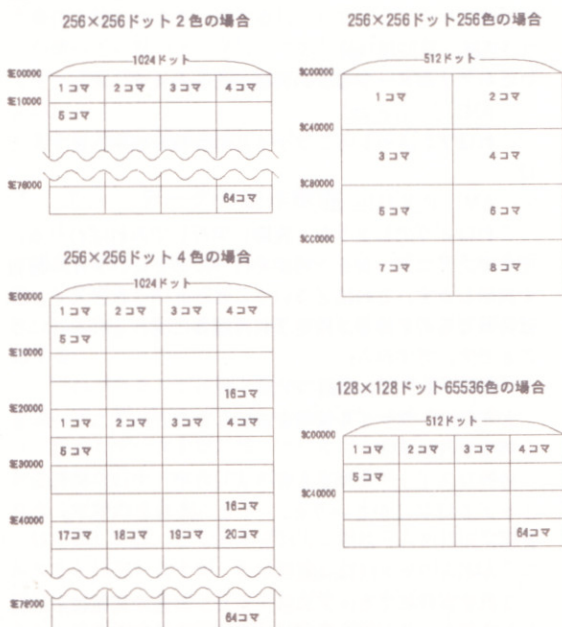
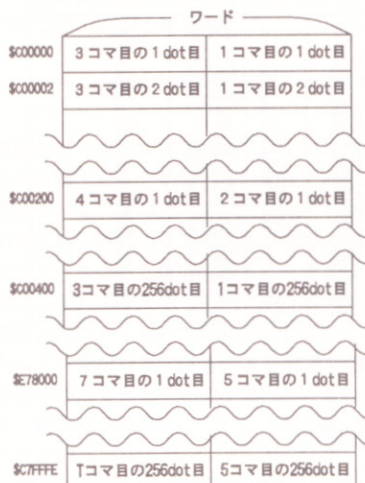


図7.2 256×256ドット256色データの並び



7. 4. 再生モード&再生能力

では、現在のSCSI装置ではどれほどの再生能力を実現することができるのでしょうか。それは、再生モードによってばらつきがあるので、まずは再生モードの解説から入りたいと思います。

いまのところ再生モードは19種をサポートしています。とりあえず表1を見てください。左から再生モード、パレット、色数、ドット数、1フレーム中のコマ数となっています。再生モードの数値が連続していないのは、その数値を2進数で表したときに、それぞれのビットに意味を持たせるためです。各ビットの意味については図7.3を見てください。空いている再生モードは将来拡張されるかもしれません。

パレットには固定と可変とがあります。固定というのは、ひとつのアニメーションデータ全体で同一のパレットを使用することを意味し、可変というのは、1コマずつに別パレットを用意しておくことを意味します。

色数とドット数の意味はわかると思います。可変パレットモードのドット数が縦に少し潰れているのは、その部分にパレットデータが収まるためです。

1フレーム中のコマ数というのは、AMIシステム上の基本データ単位であるフレームデータ（512Kバイト固定）中に、何コマ分のデータが納まっているかという意味です。一般的にコマ=フレームと解釈するのが普通ですが、AMIシステムではそれらが別々の意味を持ちますので、注意してください。

で、やっと再生能力の話に戻ります。とりあえず私が所有するMOドライブ(SONYのRMO-S350)の最速値を表7.2に示しました。このMOドライブはいまあるMOドライブの中ではかなり遅めのドライブ（信頼性は高いので、私は気にいっている。）なので、機種によっては最速値がもう少し上がるでしょう。リアルタイムデータリードにしては、なかなかのスピードだと思います。230MバイトタイプのMOの場合、データの格納領域によって速度が変わってきますので注意してください。ディスクの後ろに行けば行くほど速くなります。

表7.1

| モード(16進) | パレット | 色数 | ドット数 | コマ数/フレーム |
|-----------|------|-------|---------|----------|
| 1 (\$01) | 固定 | 2 | 128x128 | 256 |
| 2 (\$02) | 固定 | 2 | 256x256 | 64 |
| 3 (\$03) | 固定 | 2 | 512x512 | 16 |
| 5 (\$05) | 固定 | 4 | 128x128 | 128 |
| 6 (\$06) | 固定 | 4 | 256x256 | 32 |
| 7 (\$07) | 固定 | 4 | 512x512 | 8 |
| 8 (\$08) | 固定 | 16 | 64x64 | 256 |
| 9 (\$09) | 固定 | 16 | 128x128 | 64 |
| 10 (\$0a) | 固定 | 16 | 256x256 | 16 |
| 12 (\$0c) | 固定 | 256 | 64x64 | 128 |
| 13 (\$0d) | 固定 | 256 | 128x128 | 32 |
| 14 (\$0e) | 固定 | 256 | 256x256 | 8 |
| 16 (\$10) | 固定 | 65536 | 64x64 | 64 |
| 17 (\$11) | 固定 | 65536 | 128x128 | 16 |
| 18 (\$12) | 固定 | 65536 | 256x256 | 4 |
| 41 (\$29) | 可変 | 16 | 128x127 | 64 |
| 42 (\$2a) | 可変 | 16 | 256x255 | 16 |
| 45 (\$2d) | 可変 | 256 | 128x124 | 32 |
| 46 (\$2e) | 可変 | 256 | 256x254 | 8 |

図7.3 再生モードビットの意味



表7.2 RMO-S350最速値表

| モード(16進) | 最速値 [コマ/sec] | 同期カウンタ |
|-----------|--------------|--------|
| 1 (\$01) | 60 | 1 |
| 2 (\$02) | 60 | 1 |
| 3 (\$03) | 15 | 4 |
| 5 (\$05) | 60 | 1 |
| 6 (\$06) | 30 | 2 |
| 7 (\$07) | 8.6 | 7 |
| 8 (\$08) | 60 | 1 |
| 9 (\$09) | 60 | 1 |
| 10 (\$0a) | 15 | 4 |
| 12 (\$0c) | 60 | 1 |
| 13 (\$0d) | 30 | 2 |
| 14 (\$0e) | 8.6 | 7 |
| 16 (\$10) | 60 | 1 |
| 17 (\$11) | 15 | 4 |
| 18 (\$12) | 4.3 | 14 |
| 41 (\$29) | 60 | 1 |
| 42 (\$2a) | 15 | 4 |
| 45 (\$2d) | 30 | 2 |
| 46 (\$2e) | 8.6 | 7 |

最大再生時間については、1フレームデータを512Kバイトとし、1フレーム中のコマ数から1コマの容量を計算し、全体の容量から全コマ数を計算し、秒間表示コマ数で割れば計算できます。秒間表示コマ数を最速値にして、120Mバイトのデータを作ると、ほとんどのモードで、4分強の再生時間となります。

7. 5. AMIで使用するデータファイル

基本的に、完成したAMIのアニメーションデータは1個のファイルになります(通常、拡張子は[.ami]を使用)。そのほかに、パレットデータを収めるパレットファイル(通常、拡張子は[.pal]を使用)、AMIの基本データ単位であるフレームデータを収めるフレームファイル(通常、拡張子は[.amf]を使用、ファイルサイズが512Kバイト固定)、などを扱います。

パレットファイルとフレームファイルに関しての注意事項があります。パレットファイルとフレームファイルには、そのファイルがどの再生モード用のデータなのかという情

報は含まれていません。それでも、パレットデータなら、ファイルサイズから何色パレットなのか判断できますが、フレームファイルは、それ単体でどの再生モードのデータなのか判断するのは非常に難しいです(というより、どの再生モードのデータにもなり得る構造をしている)。ですから、ファイルネームを工夫するなどして各自で管理してください。

さらにもうひとつ注意しなければならないことがあります。これはAMIファイルの最大の欠点になってしまっている問題なのですが、AMIファイルを正常に(処理落ちなく)再生しようとする場合、AMIファイルが連続セクタ上にあるという条件を満たさなくてはなりません。これは、データがディスクの不連続セクタ上に存在した場合、データの読み込み時に、シーク動作によるロスタイムが生じてしまうことに起因しています。これを回避する簡易再生モードも加えてはおきましたが、なるべく一度書いたファイルはなるべく消さないとか、フリーウェアのrefreshgを持っているのなら、こまめに実行するなどの処置を行ってください。

使用セクタが不連続なファイルが再生ができないということはありませんが、AMIの性能を十分に引き出すことはできません。どうしてもという場合は、不連続セクタ上のファイルを再生する簡易再生モードがあるので、処理落ちはするものの一応見ることはできます。あらかじめ、秒間表示コマ数を少なめに設定しておけば、簡易再生モードでも処理落ちなく再生することは可能です。

7. 6. 実行ファイル

●AMI.x

アニメーションの再生はもちろん、各種設定からパレットやフレームの出し入れまで、AMIファイルに対する操作はすべてこの実行ファイルで行います。

細かい操作方法は別項を参照してもらうとして、ここではAMI.xの具体的な使用例をいくつか挙げてみたいと思います。

まず、

AMI -P file.ami

これでアニメーションの再生が行われます。そして、

AMI -V file.ami

これはアニメーションファイルの情報を表示します。では、

AMI -P-V file.ami

これはどうでしょうか、実際に実行してみればわかるのですが、アニメーション再生を行ったあとにファイル情報を表示します。これはどういうことかという、スイッチは特殊なものを除き、指定された順番に実行されるということです。ですから、

AMI -V-P file.ami

とすれば、ファイル情報を表示したあと、アニメーション再生を行います。

特殊なスイッチを除きと書きましたが、では、特殊なスイッチとはなにかというと、-?と-Mの2つです。たとえば、

AMI -P-V-? file.ami

これを実行しても、アニメーション再生や情報表示は行われません。ただ、ヘルプメッセージを表示するだけです。

また、

AMI-M-P-V file.ami

これも同じで、実行結果は、再生モード表を表示するだけです。

これで、だいたいの操作方法はわかってもらえたと思います。あとは、実際にいじってみてください。

●AMIFE.x

これは、フレームデータからコマデータを出し入れするための実行ファイルです。

またも、実行例で示しましょう。

AMIFE-G10, 3 file.amf file.pal

これは、フレームファイルから再生モード10番のコマ番号が3のコマデータを取り出しパレットファイルに従ってVRAMに展開します。もちろん、フレームファイル、パレットファイルは再生モード10番用のデータでなければなりません。次に、

AMIFE-G18, 2 file.amf

とすると、同様にコマデータをVRAMに展開するのですが、パレットデータが指定されていません。なぜならば、65536色モードデータと可変パレットモードのデータにはパレットファイルが必要ないからです。再生モード18番は65536色モードデータです、よってパレットファイルは必要ないわけです。

ここで注意が必要です。VRAMに展開されたデータは、指定した再生モードにかかわらず、65536色モードデータに変換されます。これは、現在X680x0用の多くのグラフィックツールが65536色モード用であるということを考慮したうえでの措置です。

AMIFE-S9, 4 file.amf file.pal

今度は逆にVRAMのデータをフレームファイルに書き込むわけです。VRAMへの展開と同じように、再生モードによってはパレットは必要ありません。

書き込み時も読み込み時と同様にVRAM上のデータは65536色モードのデータでなければなりません、さらに書き込み時には、「VRAM上のデータが指定した再生モードデータに変換可能な状態になっていなければならない」という条件も加わります。要するに、パレットファイル内の色しか使っていないデータにしておかなければならないわけです（可変パレットの場合は使用色数だけ制限してあればよい）。

あとはAMI.x同様実際にいじってみてください。

7. 7. データの制作環境

現在AMIシステム用のアニメーションデータを作る手段として考えられるのは、

- 1) 65536色グラフィックエディタを使って1コマずつ作画する
 - 2) プログラミング技術と数学的頭脳と膨大な計算時間を駆使して、X680x0にアニメーションデータを作らせる
 - 3) 豊富なデータを既に持っているほかのアニメーションシステムからデータのコンバートをする
- 以上3点があります。

1)に関しては、とりあえず努力と根性さえあればなんとかなると思います。私は遠慮したいと思います。2)に関してはいくつかサンプルプログラムを用意しました。しかし、まだ膨大な計算時間というのが残っていますので、

暇そうなX68000が近くにいたらぜひやらせて見てください（サンプルプログラムの説明は別記）。そして、残ったのが3)です（無理矢理な展開）。

7. 8. DoGAデータのコンバート

豊富なデータをすでに持っているアニメーションシステムといえば、なんといってもDoGAシステムでしょう。とりあえず、ここではDoGAのPICファイルがすでに用意してあるという前提の下で話を進めます。

コンバートなどとはいっても、やっтерことは単純で、1枚DoGA PICを表示してはAMIFEでフレームファイルに書き込んでいくという単純な繰り返しをバッチファイルで行っているだけです。リスト1を見てください。いっておきますが、これをそのまま実行しても動く保証はありません、各自自分の環境にあわせて書き直したあとで実行してください。

まず、1～5行で各種初期化を行っています。AMIファイルは再生モード45で初期化しています。AMIENV.xというのは環境変数を操作する小物ツールで、3行目で、環境変数komaに000が、4行目で、環境変数picに001がセットされます。

7～16行がメインループです。8～10行で実際のコンバートを行っています。slide.xでDoGA PICを表示して、AMIZM.xという小物ツールで、256×256の画像を512×512へ拡大したあと、AMIFE.xで1コマ、フレームファイルに書き込みます。

12～13行は環境変数komaをインクリメントしたあと、32（1フレーム中のコマ数）と比較して、もし32になったらkomaを初期値に戻し、フレームファイルをAMIファイルに書き足しています。

15～16行はループ判定です。環境変数picを、DoGA PICの枚数+1と比較しています。

18行以降はDoG APICの枚数が32で割り切れないときの処理で、端数分を最後のDoGA PICデータで埋めています。

これで、ほとんどのDoGAデータはAMIデータにコンバートできると思います。

IF文など、バッチファイル用の内部コマンドに関しては、Human68kのマニュアル、slide.xに関してはDoGAのマニュアルを参照してください。

7. 9. 小物ツール

●AMIENV.x

3桁の10進文字列を環境変数で扱うツールで、スイッチ-C[n]でnに初期化、-A[n]で+nして、-S[n]で-nします。

●AMIZM.x

65536色モードのグラフィックを拡大するもので、デフォルトが256×256から512×512への拡大、スイッチで128×128、64×64から512×512も指定できます。

●AMIPC.x

16色モード、または256色モードのグラフィックを65536色モードへ変換します。スイッチ-Pでページ指定ができます。

●toG3R3B2.x

65536色モードデータをグリーン3ビット、レッド3ビ

リスト1

```
1: echo off
2: screen 1,3,1
3: amienv -c0 koma
4: amienv -cl pic
5: ami b:\file.ami -i45,3
6:
7: :LOOP1
8: slide DoGA%pic%.pic
9: amizm -256
10: amife %temp%file.amf -s45,%koma%
11:
12: amienv -al koma
13: if %koma% == 032 amienv -c0 koma||ami b:\file.
ami -ra %temp%file.amf
14:
15: amienv -al pic
16: if not %pic% == 041 goto LOOP1
17:
18: if %koma% == 000 goto END
19: amienv -sl %koma%
20: amife %temp%file.amf -g45,%koma%
21: amienv -al %koma%
22:
23: :LOOP2
24: amife %temp%file.amf -s45,%koma%
25:
26: amienv -al koma
27: if not %koma% == 032 goto LOOP2
28: ami b:\file.ami -ra %temp%file.amf
29:
30: :END
```

ット、ブルー2ビットだけを使用した256色に減色します
(ただし、グラフィックモードは65536色モードのまゝ)。

●toTONE16.x

toG3R3B2.xと同様に、ものトーン16階調に減色します。

7. 10. AMIシステム用ツール開発環境

一応、Cもしくはアセンブラで使用可能なライブラリを用意しました。もし、AMIシステムに少しでも興味を持って、「なにかツールでも作って見ようかな」と思った方がいましたら、ぜひともチャレンジしてみてください、よろしくをお願いします。

開発用のファイルは、ライブラリ本体のAMILIB.l、C用ヘッダファイルのAMILIB.h、アセンブラ用マクロファイルのAMILIB.macがあります。その他にはAMI.xやAMIFE.xのソースファイルが参考になるかもしれません。

ライブラリの各関数の説明はAMILIB.hの中に関数のプロタイプ宣言と共に示してあります。

7. 11. AMIファイルの構造

AMIファイルは大きく分けて、2つのブロックで構成されています。それぞれ、ヘッダブロックとフレームデータブロックと呼びます。図2に簡単な図解を示しました。

AMIファイルの先頭1024バイトが、ヘッダブロックです。その内容は、識別シンボル、コメント、再生モード、同期カウンタ、再生フレーム数、パレット、空きとなっています。

識別シンボルとはそのファイルがAMIファイルかどうか識別するためのデータで、8バイトの定数で示してあります。

コメントは全角で60文字まで書くことができます。新たにデータを制作した場合、なるべくコメントをつけるようにしてください。

再生モードは、AMIファイルがどの再生モードデータなのかを示しています。

同期カウンタは、再生スピードを表すデータで、1コマを垂直同期何回分表示するかで表します。たとえば、同期カウンタが4と設定されていたら、1コマを垂直同期4回分表示するので、1秒間に15コマの表示スピードになります(垂直同期は秒間60回)。

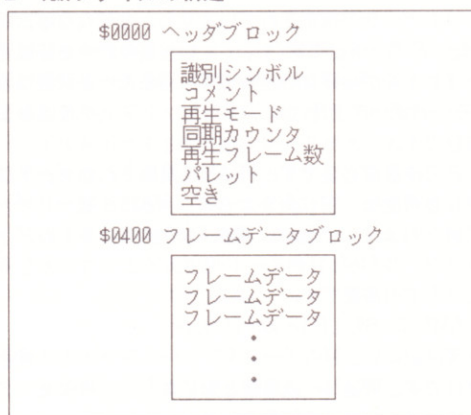
再生フレーム数は、再生するフレーム数であって、実際にAMIファイルに含まれているフレーム数ではありませんので注意してください。

パレットは固定パレットのデータのときに参照されます。

空きは将来の拡張用です。通常0で埋められています。

ヘッダブロックの後にはフレームデータブロックがあります。フレームデータブロックはフレームデータが複数連なった構造をしています。すで書いたように、フレームデータブロックに連なっているブロック数は、ヘッダブロックの再生フレーム数とは違いますので注意してください。

図2 AMIファイルの構造



EX-Systemコール一覧

ここでは、EX-System ver. 3.0でサポートされるEXコールのC言語用ライブラリ関数を、アルファベット順に掲載しています。ここに掲載された関数を使用する場合には、プログラム内でEXLIB.Hをインクルードし、コンパイル時にEXLIB.Lをリンクしなければなりません。

ACTPAL \$1A

【書式】

```
char ACTPAL(palno);
char palno; /*パレット番号*/
```

【機能】

アクティブパレット番号を設定します。

palno=-1で現在のアクティブパレット番号を得ます。

【戻り値】

以前のアクティブパレット番号

【参照関数】

SETCOL, GETCOL

ALTERNATE \$0A

【書式】

```
int ALTERNATE(md);
int md; /*モード*/
```

【機能】

モード番号=0のとき

作業画面と裏画面を入れ換えます。

モード番号=1のとき

裏画面を作業画面にコピーします。

【戻り値】

0: 正常終了

-1: 裏画面使用不可

BUF2GRAM \$05

【書式】

```
void BUF2GRAM(square);
struct SQUARE *square;
struct SQUARE {
    int x1; /*始点X座標*/
    int y1; /*始点Y座標*/
    int x2; /*終点X座標*/
    int y2; /*終点Y座標*/
};
```

【機能】

指定した領域を待避画面より復帰します。

また、square=NULLで画面全体を復帰します。

【戻り値】

なし

【参照関数】

GRAM2BUF, COMGVRAM, BUF2GRAM2

BUF2GRAM2 \$2D

【書式】

```
void BUF2GRAM2(n);
```

```
int n; /*マスク表示フラグ*/
```

【機能】

ウィンドウを除く画面全体を待避画面から表示画面に転送します。

マスク表示フラグが1のときはマスクを表示しません。

【戻り値】

なし

【参照関数】

GRAM2BUF, BUF2GRAM, COMGVRAM

BUFFADR \$1F

【書式】

```
unsigned short *BUFFADR0;
```

【機能】

待避画面の先頭アドレスを得ます。

【戻り値】

待避画面の先頭アドレス

【参照関数】

GETADR, MASKADR

CLOSEWIN \$14

【書式】

```
void CLOSEWIN0;
```

【機能】

WINITEMにより送られた情報に従ってウィンドウをクローズします。

【戻り値】

なし

【参照関数】

WINITEM, OPENWIN

COMGVRAM \$1E

【書式】

```
void COMGVRAM(square);
struct SQUARE *square;
struct SQUARE {
    int x1; /*始点X座標*/
    int y1; /*始点Y座標*/
    int x2; /*終点X座標*/
    int y2; /*終点Y座標*/
};
```

【機能】

指定した領域のG-RAMに書き込まれた内容と待避画面の内容を合成し、それぞれに反映します。特にG-RAMのデータが\$F801の場合はマスクがクリアされます。

また、square=NULLで画面全体を対象とします。

【戻り値】

なし

【参照関数】

GRAM2BUF, BUF2GRAM, BUF2GRAM2

CONFIRM \$01

【書式】

```
void CONFIRM(msgptr);
char *msgptr; /*文字列へのポインタ*/
```

【機能】

確認ウィンドウをオープンします。DOSコール PRINT

でも同じ動作をします。

[戻り値]

なし

[参照関数]

SELECT

DRAW \$26

[書式]

```
void DRAW(x, y, adr);
int x; /* 描画開始X座標
int y; /* 描画開始Y座標
void *adr; /* 描画中に呼び出す関数へのアドレス
```

[機能]

現在のドローモード構造体で示されているモードで、描画を行います。

描画中に関数を呼ばない場合は、adr を NULL にしてください。

[戻り値]

なし

[参照関数]

GETDRAWMODE

EXNUM \$20

[書式]

```
int EXNUM0;
```

[機能]

動作中の EX-System のナンバーを返します。

[戻り値]

0: Zs-EX
1: Matier-EX
2: EX-Window

[参照関数]

VERSION

FILEWIN \$03

[書式]

```
int FILEWIN(wintpr);
struct WINPTR *wintpr;
struct WINPTR {
    char *title; /* タイトル名へのポインタ */
    char *ftype1; /* ファイルタイプ名 1 へのポインタ */
    char *fname1; /* 表示ファイル名 1 へのポインタ */
    char *ftype2; /* ファイルタイプ名 2 へのポインタ */
    char *fname2; /* 表示ファイル名 2 へのポインタ */
    char *fname; /* 選択ファイル名格納領域へのポインタ */
};
```

[機能]

ファイルウィンドウをオープンします。

表示ファイル名は必ず "*" で始めてください。また、ファイルタイプがひとつのときはファイルタイプ名 2 を NULL にしてください。メンバ fname は内部の静的な領域へのポインタを示します。したがって、FILEWIN が呼び出されるたびに同じ領域に上書きされ前の内容は失われます。

[戻り値]

-1: ESC キーでウィンドウクローズ
0: クローズボックスでウィンドウクローズ

1: ファイルタイプ 1 でロード

2: ファイルタイプ 1 でセーブ

3: ファイルタイプ 2 でロード

4: ファイルタイプ 2 でセーブ

FIX \$2E

[書式]

```
void FIX0;
```

[機能]

待避画面をアンドウバッファに転送します。

[戻り値]

なし

[参照関数]

UNDO, UNDOBUFFADR, UNDOMASKADR

GETADR \$00

[書式]

```
unsigned short *GETADR0;
```

[機能]

裏画面の先頭アドレスを得ます。外部ファイルとしてプログラムが呼ばれたときの第 1 引数と同じ値です。

[戻り値]

裏画面の先頭アドレス

[参照関数]

MASKADR, BUFFADR

GETAREA \$0C

[書式]

```
int GETAREA(square);
struct SQUARE *square;
struct SQUARE {
    int x1; /* 始点X座標 */
    int y1; /* 始点Y座標 */
    int x2; /* 終点X座標 */
    int y2; /* 終点Y座標 */
};
```

[機能]

矩形領域の指定を行わせ、その座標を得ます。

[戻り値]

0: 正常終了
-1: マウス右クリックによる強制終了

[参照関数]

GETPOINT

GETCOL \$19

[書式]

```
unsigned short GETCOL(palno);
char palno; /* パレット番号 */
```

[機能]

システムパレットの色コードを得ます。

[戻り値]

パレット番号 palno の色コード

[参照関数]

SETCOL, ACTPAL

GETDRAWMODE \$24

[書式]

```

sDrawMode      *GETDRAWMODE0;
typedef struct sDrawMode { /*描画モード構造体*/
    unsigned short Color; /*カレントカラー*/
    unsigned char Mask; /*カレントマスク*/
    eDrawKind Draw; /*描画種類*/
    int Pen; /*ペン番号0でドットペン*/
    unsigned char *Pattern; /*カレントペンパターン*/
    /*16×16のパターンへのポインタ*/
    eDrawSource Source; /*描画ソース*/
    eEditObject Object; /*編集対象*/
    eBorder Border; /*ベイント境界*/
    BOOL BorderFlag; /*境界指定フラグ*/
    unsigned short BorderColor; /*境界カラー*/
    unsigned char BorderMask; /*境界マスク*/
    BOOL ExchangeFlag; /*色変換フラグ*/
    unsigned short ColorGrad[4];
    /*グラデーションカラー 左上 右上 左下 右下の順*/
    unsigned char MaskGrad[4];
    /*マスクグラデーション*/
    unsigned short Change[2][12];
    /*色変換設定パレット*/
} sDrawMode;
typedef enum eDrawKind { /*描画種類*/
    eDrawKind_POINT, /*点*/
    eDrawKind_LINE, /*直線*/
    eDrawKind_BOX, /*矩形*/
    eDrawKind_BOXFILL, /*塗りつぶし矩形*/
    eDrawKind_ELLIPSE, /*楕円*/
    eDrawKind_ELLIPSEFILL, /*塗りつぶし楕円*/
    eDrawKind_PAINT, /*ベイント*/
    eDrawKind_GRADATION, /*グラデーションフィル*/
    eDrawKind_COPY, /*コピー*/
    eDrawKind_RESIZE, /*リサイズコピー*/
    eDrawKind_REVERSE, /*色反転*/
    eDrawKind_MIRROR, /*左右反転*/
    eDrawKind_FLIP, /*上下反転*/
    eDrawKind_CHANGE, /*色変換*/
    eDrawKind_MAX, /*描画種類数*/
} eDrawKind;
typedef enum eDrawSource { /*描画ソース*/
    eDrawSource_COLOR, /*カレントカラー*/
    eDrawSource_MASK, /*カレントマスク*/
    eDrawSource_REMOVE, /*カレントマスクによるマスク削除*/
    eDrawSource_ALTERNATECOLOR, /*裏画面カラー*/
    eDrawSource_ALTERNATEMASK, /*裏画面マスク*/
} eDrawSource;
typedef enum eEditObject { /*編集対象*/
    eEditObject_COLOR, /*カラー*/
    eEditObject_MASK, /*マスク*/
    eEditObject_ALL, /*すべて*/
} eEditObject;
typedef enum eBorder { /*ベイント境界*/
    eBorder_COLOR, /*カラー*/
    eBorder_MASK, /*マスク*/

```

```

    eBorder_ALL, /*すべて*/
} eBorder;
typedef enum BOOL {
    FALSE, /*偽*/
    TRUE, /*真*/
} BOOL;

```

[機能]

現在のドローモードを示す構造体へのアドレスを得ます。

[戻り値]

ドローモード構造体へのアドレス

[参照関数]

DRAW

GETEXISTWINDOW \$25

[書式]

```

sExistWindow *GETEXISTWINDOW0;
typedef struct sExistWindow { /*ウィンドウ存在情報*/
    int x1; /*ウィンドウ左 X座標*/
    int y1; /*ウィンドウ上 Y座標*/
    int x2; /*ウィンドウ右 X座標*/
    int y2; /*ウィンドウ下 Y座標*/
    struct sExistWindow *next;
    /*次のウィンドウ座標構造体へのポインタ*/
    /*なければNULL*/
} sExistWindow;

```

[機能]

現在表示されているウィンドウ情報を示す構造体へのアドレスを得ます。

[戻り値]

ウィンドウ情報構造体へのアドレス

GETLUPEWINDOW \$28

[書式]

```

sLupeWindow *GETLUPEWINDOW(n);
int n; /*ルーペNo. 0 or 1*/
typedef struct sLupeWindow {
    int x1; /*被拡大領域左 X座標*/
    int y1; /*被拡大領域上 Y座標*/
    int x2; /*被拡大領域右 X座標*/
    int y2; /*被拡大領域下 Y座標*/
    int wx1; /*拡大領域左 X座標*/
    int wy1; /*拡大領域上 Y座標*/
    int wx2; /*拡大領域右 X座標*/
    int wy2; /*拡大領域下 Y座標*/
    int f; /*拡大率*/
} sLupeWindow;

```

[機能]

ルーペ情報を取得します。

ルーペ情報構造体メンバのうち、

$wx2-wx1+1 = (x2-x1+1) \times f$

$wy2-wy1+1 = (y2-y1+1) \times f$

の関係が成り立ちます。

[戻り値]

ルーペ情報へのアドレス

[参照関数]

SETLUPEWINDOW, PAINTLUPE, LUPE2SCREEN, L
UPEPSET, LUPEREVSET

GETPOINT \$0D

[書式]

```
int GETPOINT(x, y);
int *x; /* X座標格納領域へのポインタ */
int *y; /* Y座標格納領域へのポインタ */
```

[機能]

点の指定を行わせ、その座標を得ます。

[戻り値]

0: 正常終了
-1: マウス右クリックによる強制終了

[参照関数]

GETAREA

GETSYSTEMMODE \$30

[書式]

```
sSystemMode *GETSYSTEMMODE0;
typedef struct sSystemMode {
    int DoubleClick; /* ダブルクリックスピード */
    BOOL FullDrag; /* フルドラッグ */
    BOOL Aspect; /* アスペクト比1:1モード
                  (Zs-EX無効) */
} sSystemMode;
```

[機能]

現在のシステム情報を示す構造体へのアドレスを得ます。

[戻り値]

システム情報構造体へのアドレス

[参照関数]

SETASPECT

GPAINT \$0B

[書式]

```
void GPAINT(x, y, c);
int x; /* X座標 */
int y; /* Y座標 */
unsigned short c; /* カラーコード */
```

[機能]

ペイントをします。

[戻り値]

なし

[参照関数]

GPAINT_M

GPAINT_M \$21

[書式]

```
void GPAINT_M(x, y, c);
int x; /* X座標 */
int y; /* Y座標 */
unsigned short c; /* カラーコード */
```

[機能]

マスクを境界としてペイントをします。

[戻り値]

なし

[参照関数]

GPAINT

GRAM2BUF \$04

[書式]

```
void GRAM2BUF(square);
struct SQUARE *square;
struct SQUARE {
    int x1; /* 始点X座標 */
    int y1; /* 始点Y座標 */
    int x2; /* 終点X座標 */
    int y2; /* 終点Y座標 */
};
```

[機能]

指定した領域を待避画面に待避します。

また、square=NULLで画面全体を待避します。

[戻り値]

なし

[参照関数]

BUF2GRAM, COMGVRAM, BUF2GRAM2

LEDIT \$23

[書式]

```
int LEDIT(x, y, l, lo, str);
int x; /* 編集セルの左上X座標 */
int y; /* 編集セルの左上Y座標 */
int l; /* 編集セルの文字幅 */
int lo; /* 文字列の最大長 (含ヌル) */
char *s; /* 文字列へのポインタ */
```

[機能]

与えられた文字列に対して、1行編集を行います。

[戻り値]

0: 正常終了
1: ESCキーによるキャンセル

LUPE2SCREEN \$2A

[書式]

```
void LUPE2SCREEN(x, y);
int *x; /* ルーベ内X座標 */
int *y; /* ルーベ内Y座標 */
```

[機能]

ルーベ内座標を実座標に変換し、*x, *yに格納します。

座標がルーベ外のときは変換しません。

[戻り値]

なし

[参照関数]

SETLUPEWINDOW, GETLUPEWINDOW, PAINTLUP
E, LUPEPSET, LUPEREVSET

LUPEPSET \$2B

[書式]

```
void LUPEPSET(x, y, c);
int x; /* 実X座標 */
int y; /* 実Y座標 */
unsigned short c; /* カラー */
```

[機能]

ルーベ内に点を描画します。作業画面や表示画面には反映されません。

[戻り値]

なし

[参照関数]

SETLUPEWINDOW, GETLUPEWINDOW, PAINTLUP
E, LUPE2SCREEN, LUPEREVSET

LUPEREVSET \$2C

[書式]

```
void      LUPEREVSET(x, y);  
int       x;      /* 実 X 座標 */  
int       y;      /* 実 Y 座標 */
```

[機能]

ルーベ内に反転色で点を描画します。作業画面や表示画面には反映されません。

[戻り値]

なし

[参照関数]

SETLUPEWINDOW, GETLUPEWINDOW, PAINTLUP
E, LUPE2SCREEN, LUPEPSET

MANAGE \$12

[書式]

```
int      MANAGE(n);  
int      n;      /* 動作モード */
```

[機能]

WINITEMにより送られた情報に従って選択状態を返します。

動作モードのビット0が0のときはリアルタイムで返し、1のときはアイテムが選択されるまで待ちます。ビット1が1のときはスペースキーによる裏表画面反転、XF1キーによるマスク透視、登録キーによるリドゥおよびUNDOキーによるアンドゥを許します。

[戻り値]

000FLRLRNNNNNNNN

ビット12

F: マウスがアイテム上にないときのフラグ

ビット11・9

LL: 左ボタン

(0:off, 1:Single, 2:Double, 3:Triple)

ビット10・8

RR: 右ボタン

(0:off, 1:Single, 2:Double, 3:Triple)

ビット7～0 NNNNNNNN: アイテム番号(0～255) 特に255のときはESCキー

WINITEMで送った itemptr.x0 と itemptr.y0 にウィンドウの左上座標を返します。

スペースキーによる表裏画面反転が行われたときにはアイテム番号254, XF1キーによるマスク透視およびUNDOキーによるアンドゥを行ったときにはアイテム番号253を返します。

[参照関数]

WINITEM, OPENWIN

MASKADR \$1D

[書式]

```
unsigned char *MASKADR(page);  
int          page;      /* ページ */
```

[機能]

8ビットマスクバッファの先頭アドレスを得ます。

page=0で作業画面, 1で裏画面のマスクバッファに対応します。

[戻り値]

マスクバッファの先頭アドレス

0なら8ビットマスク使用不可

[参照関数]

GETADR, BUFFADR

MCSET \$09

[書式]

```
void      MCSET(no);  
int       no;      /* カーソル番号 */
```

[機能]

マウスカーソルを選択します。

0: READY

1: BUSY

2: SPUIT

[戻り値]

なし

MINT \$22

[書式]

```
void      MINT(mod);  
int       mod;      /* モード */
```

[機能]

マスク・マウス割り込みの状態を変更します。

モード=1で割り込み許可, 0で禁止です。

[戻り値]

なし

MOVEWIN \$13

[書式]

```
void      MOVEWIN(x, y);  
int       x;      /* 現在のマウスカーソルのX座標 */  
int       y;      /* 現在のマウスカーソルのY座標 */
```

[機能]

WINITEMにより送られた情報に従って、ウィンドウをマウスの左ドラッグにあわせて移動させます。

[戻り値]

WINITEMで送った itemptr.x0 と itemptr.y0 にウィンドウの左上座標を返します。

[参照関数]

WINITEM, OPENWIN, TRANSWIN

OPENWIN \$11

[書式]

```
void      OPENWIN(titleptr);  
char      *titleptr; /* タイトル名へのポインタ */
```

[機能]

WINITEMにより送られた情報に従ってウィンドウをオープンし、ウィンドウタイトルを表示します。

WINITEMで itemptr.x0 および itemptr.y0 を-1として送った場合は、以前に表示されていたウィンドウの左上座標が与えられます。

[戻り値]

WINITEMで itemptr.x0 および itemptr.y0 を-1として

送った場合は、itemptr.x0およびitemptr.y0に以前に表示されていたウィンドウの左上座標を返します。

また、ウィンドウが画面からはみ出す場合は、itemptr.x0およびitemptr.y0を修正して返します。

[参照関数]

WINITEM, MANAGE

PAINTLUPE \$29

[書式]

```
void PAINTLUPE(n);
int n; /* ルーベNo. 0 or 1 */
```

[機能]

ルーベ内を描画します。

[戻り値]

なし

[参照関数]

SETLUPEWINDOW, GETLUPEWINDOW, LUPE2SCREEN, LUPEPSET, LUPEREVSET

REVBOX \$07

[書式]

```
void REVBOX(square);
struct SQUARE square;
struct SQUARE {
    int x1; /* 始点X座標 */
    int y1; /* 始点Y座標 */
    int x2; /* 終点X座標 */
    int y2; /* 終点Y座標 */
};
```

[機能]

ボックスを反転色で描画します。

[戻り値]

なし

[参照関数]

REVLIN, REVFILL

REVFILL \$08

[書式]

```
void REVFILL(square);
struct SQUARE *square;
struct SQUARE {
    int x1; /* 始点X座標 */
    int y1; /* 始点Y座標 */
    int x2; /* 終点X座標 */
    int y2; /* 終点Y座標 */
};
```

[機能]

ボックスフィルを反転色で描画します。

[戻り値]

なし

[参照関数]

REVLIN, REVFILL

REVLIN \$06

[書式]

```
void REVLIN(square);
struct SQUARE *square;
```

```
struct SQUARE {
    int x1; /* 始点X座標 */
    int y1; /* 始点Y座標 */
    int x2; /* 終点X座標 */
    int y2; /* 終点Y座標 */
};
```

[機能]

ラインを反転色で描画します。

[戻り値]

なし

[参照関数]

REVB, REVFILL

ROLLDOWN \$0F

[書式]

```
void ROLLDOWN(rollptr);
struct ROLLPTR *rollptr;
struct ROLLPTR {
    int x0; /* 左上X座標 */
    int y0; /* 左上Y座標 */
    int x; /* Xサイズ */
    int y; /* Yサイズ */
    int d; /* スクロールドット数 */
    unsigned short c; /* 下地カラー */
};
```

[機能]

指定領域をdドット下にスクロールします。

[戻り値]

なし

[参照関数]

ROLLUP

ROLLUP \$0E

[書式]

```
void ROLLUP(rollptr);
struct ROLLPTR *rollptr;
struct ROLLPTR {
    int x0; /* 左上X座標 */
    int y0; /* 左上Y座標 */
    int x; /* Xサイズ */
    int y; /* Yサイズ */
    int d; /* スクロールドット数 */
    unsigned short c; /* 下地カラー */
};
```

[機能]

指定領域をdドット上にスクロールします。

[戻り値]

なし

[参照関数]

ROLLDOWN

SELECT \$02

[書式]

```
int SELECT(msgptr);
char *msgptr; /* 文字列へのポインタ */
```

[機能]

選択ウィンドウをオープンします。IOCSコール B_PRI

NT でも同じ動作をします。

[戻り値]

0: OK

1: CANCEL

[参照関数]

CONFIRM

SETASPECT \$34

[書式]

BOOL SETASPECT(mode);

BOOL mode; /*アスペクトモード*/

[機能]

アスペクト比の設定を行います。

Zs-EXでは機能しません。

アスペクトモードがTRUE のときドットアスペクト比を1:1にします。FALSE のときドットアスペクト比を4:3にします。

[戻り値]

以前のアスペクトモード

[参照関数]

GETSYSTEMMODE

SETCOL \$18

[書式]

unsigned short SETCOL(palno, col);

char palno; /*パレット番号*/

unsigned short col; /*色コード*/

[機能]

システムパレットに色コードを設定します。

[戻り値]

以前のパレット番号 palno の色コード

[参照関数]

GETCOL, ACTPAL

SETLUPEWINDOW \$27

[書式]

void SETLUPEWINDOW(n, lupe);

int n; /*ルーベNo. 0 or 1*/

sLupeWindow *lupe;
/*ルーベ情報構造体へのアドレス*/

typedef struct sLupeWindow {

int x1; /*被拡大領域左 X座標*/

int y1; /*被拡大領域上 Y座標*/

int x2; /*被拡大領域右 X座標*/

int y2; /*被拡大領域下 Y座標*/

int wx1; /*拡大領域左 X座標*/

int wy1; /*拡大領域上 Y座標*/

int wx2; /*拡大領域右 X座標*/

int wy2; /*拡大領域下 Y座標*/

int f; /*拡大率*/

} sLupeWindow;

[機能]

ルーベ情報を設定します。

ルーベ情報構造体メンバのうち、

wx2-wx1+1 = (x2-x1+1)*f

wy2-wy1+1 = (y2-y1+1)*f

の関係が成り立ちます。

[戻り値]

なし

[参照関数]

GETLUPEWINDOW, PAINTLUPE, LUPE2SCREEN, L
UPEPSET, LUPEREVSET

TRANSWIN \$16

[書式]

void TRANSWIN(x, y);

int x; /*移動先ウィンドウ左上X座標*/

int y; /*移動先ウィンドウ左上Y座標*/

[機能]

WINITEMにより送られた情報に従ってウィンドウを移動させます。

[戻り値]

WINITEMで送った itemptr.x0 と itemptr.y0 にウィンドウの左上座標、すなわち x と y を返します。

[参照関数]

WINITEM, OPENWIN, MOVEWIN

UNDO \$2F

[書式]

void UNDO();

[機能]

待避画面とアンドウバッファを入れ換えます。

[戻り値]

なし

[参照関数]

FIX, UNDOBUFFADR, UNDOMASKADR

UNDOBUFFADR \$31

[書式]

unsigned short *UNDOBUFFADR();

[機能]

アンドウ画面の先頭アドレスを得ます。

[戻り値]

アンドウ画面の先頭アドレス

0 ならアンドウ使用不可

[参照関数]

FIX, UNDO, UNDOMASKADR

UNDOMASKADR \$32

[書式]

unsigned char *UNDOMASKADR();

[機能]

アンドウマスクの先頭アドレスを得ます。

[戻り値]

アンドウマスクの先頭アドレス

0 ならアンドウまたは8ビットマスク使用不可

[参照関数]

FIX, UNDO, UNDOBUFFADR

VERSION \$17

[書式]

void VERSION();

[機能]

動作中の EX-Windows のバージョン番号を返します。

[戻り値]

整数部×256+小数部

[参照関数]

EXNUM

WINBOX

\$15

[書式]

```
void WINBOX (square);
struct SQUARE *square;
struct SQUARE {
    int x1; /* 左上絶対X座標 */
    int y1; /* 左上絶対Y座標 */
    int x2; /* 右下絶対X座標 */
    int y2; /* 右下絶対Y座標 */
};
```

[機能]

ウィンドウ内にボックスを描画します。

[戻り値]

なし

WINITEM

\$10

[書式]

```
void WINITEM (itemptr);
struct ITEMPTR *itemptr;
struct ITEMPTR {
    int x0; /* ウィンドウ左上X座標 */
    int y0; /* ウィンドウ左上Y座標 */
    int x; /* ウィンドウXサイズ */
    int y; /* ウィンドウYサイズ */
    int n; /* アイテム数 */
    ITEM *i; /* アイテムへのポインタ */
};
typedef short ITEM[10] = {
    short rev; /* 反転/クローズ/移動 */
    short ret; /* 復帰タイミング */
    short x1; /* 選択領域左上相対X座標 */
    short y1; /* 選択領域左上相対Y座標 */
    short x2; /* 選択領域右下相対X座標 */
    short y2; /* 選択領域右下相対Y座標 */
    short revx1; /* 反転領域左上相対X座標 */
    short revy1; /* 反転領域左上相対Y座標 */
    short revx2; /* 反転領域右下相対X座標 */
    short revy2; /* 反転領域右下相対Y座標 */
};
```

[機能]

ウィンドウ情報をシステムに送ります。

rev のビット 0 が 1 ならば、マウスが選択領域内にあるときに反転領域が反転します。特に 254 のときはウィンドウ移動、255 のときはウィンドウクローズを示します。

ret のビット 0 が 1 ならば、アイテムが選択されたあと、マウスのボタンを離すまで待ち、ビット 1 が 1 ならば、ダブル/トリプルクリックを有効にします。

itemptr は静的な領域で宣言してください。

[戻り値]

なし

[参照関数]

OPENWIN, MANAGE

機能別コール一覧

[バッファ操作]

ALTERNATE 裏画面と作業画面を転送/切り換える
BUF2GRAM 待避画面から表示画面に転送する
BUF2GRAM2 ウィンドウ/マスクを考慮して待避画面から表示画面に転送する
BUFFADR 待避画面の先頭アドレスを得る
CONVGRAM G-RAMと待避画面を合成する
FIX 待避画面をアンドゥバッファに転送する
GETADR 裏画面の先頭アドレスを得る
GRAM2BUF 表示画面を待避画面に転送する
MASKADR マスクバッファの先頭アドレスを得る
UNDO 待避画面とアンドゥバッファを入れ換える
UNDOBUFFADR アンドゥバッファの先頭アドレスを得る
UNDOMASKADR アンドゥマスクバッファの先頭アドレスを得る

[ウィンドウ]

CLOSEWIN ウィンドウをクローズする
CONFIRM 確認ウィンドウをオープンする
FILEWIN ファイルウィンドウをオープンする
GETEXISTWINDOW ウィンドウ情報構造体へのアドレスを得る
LEDIT 一行編集を行う
MANAGE ウィンドウのマネージメントを行う
MOVEWIN ウィンドウを移動する
OPENWIN ウィンドウをオープンする
SELECT 選択ウィンドウをオープンする
TRANSWIN ウィンドウを移動する
WINBOX ウィンドウレクタングルを描画する
WINITEM ウィンドウ情報を定義する

[G-RAM操作]

GPAINT ペイントする
GPAINT_M マスクを境界としてペイントする
REVBOX 反転色でボックスを描画する
REVFILL 反転色でボックスフィルを描画する
REVLIN 反転色でラインを描画する
ROLLODOWN 領域をスクロールダウンする
ROLLUP 領域をスクロールアップする

[マウス操作]

GETAREA 矩形領域指定を行う
GETPOINT 点指定を行う
MCSET マウスカーソル形状を変更する

[パレット]

ACTPAL アクティブパレット番号を設定する
GETCOL システムパレットの色コードを得る
SETCOL システムパレットに色コードを設定する

[ルーベ]

GETLUPEWINDOW ルーベ情報を取得する
LUPE2SCREEN ルーベ内座標を実座標に変換する
LUPESET ルーベ内に点を描画する
LUPEREVSET ルーベ内に反転色で点を描画する
PAINTLUPE ルーベ内を描画する
SETLUPEWINDOW ルーベ情報を設定する

[システム]

DRAW システムドローモードで描画を行う
EXNUM 動作中の EX のナンバーを得る
GETDRAWMODE ドローモード構造体へのアドレスを得る
GETSYSTEMMODE システム構造体へのアドレスを得る
LOADDRAW システムドロールーチンをロードする
SETASPECT ドットアスペクト比を設定する
VERSION 動作中の EX のバージョン番号を得る

[その他]

WINT マスク/マウス割り込みの状態を変更する

付録ディスクの使い方

ディスクについては、そのまま起動すればEX-Systemが立ち上がります。ただし、EX-SystemはFD上で使用することを考慮されて作成されていませんので、試用以外の目的で使うときには必ずハードディスクへ転送したうえで使用してください。頻繁にディスクアクセスを行いますので、ディスク1で設定されているAUTOEXEC.BATのようにFASTIOでディスクキャッシュを設定するとよいでしょう。

●CD-ROMの使い方

CD-ROMをアクセスするにはCD-ROMドライブとCD-ROMドライブ、Human68k ver.3.0以降が必要です。SCSI対応のCD-ROMドライブと計測技研から発売されているX68000用のCD-ROMドライブを購入してください。また、ファイル名に“-”を使用している部分がありますので、Human68k ver.2.0などではアクセスできないファイルがあります。インストールの際には必ずHuman68k ver.3.0以降をお使いください（付属のFDにはHuman68k ver.3.02が入っています）。

他機種を使用してCD-ROMを利用される方は、以下の点に注意してください。ファイル名はISO9660標準の仕様とはなっておりません（8文字+3文字ですが、一部カタカナ、漢字が入ります）。OSによっては不都合が出ることも考えられますのでご注意ください。

CD-ROM書き込みの都合上、CD-ROM内のファイルはファイル名が8文字+拡張子3文字という具合に切り詰められています。本来のファイル名でインストールするためにファイル名補完ドライバを用意しました。ルートディレクトリにあるFnResetを常駐するとCD-ROMのルートに存在するTREE.DATを参照してファイル名を本来のものにすげ替えます。データフォーマットについては簡単な技術資料がありますので、他機種版を作成する方は参考にしてください（おまけディレクトリにあります）。

CD-ROMからインストールする場合には、事前にTWENTYONEなどのロングファイルネーム対応ドライバをはずしておいてください。確実に誤動作します。

●インストールの手順

サンプルまで含めたシステムを最大限にインストールすると約96Mバイトのハードディスクを必要とします。その他のデータもインストールするとさらに大量のエリアを消費します。

推奨されるインストール例は、

SYSTEM
EFFECT
STAMP
BRUSH
TEXTURE

のディレクトリをハードディスクにインストールするというものです。この場合、約40Mバイトの容量を必要とします。ここで指定されているSTAMP、BRUSH、TEXTUREの内容はあくまでもサンプルのものです。空き容量に余裕がない場合、これらの内容はハードディスクに置く必要はありません。それぞれEX_STAMP.SYS、PATTERNBRUSH.SYS、PICPAINT.SYSで格納場所が定義されていますので、最初はCD-ROMの各ディレクトリに設定しておいて、ざっと使ってみたくて必要なものだけをハードディスクに置くとよいでしょう。ただし、その際には各設定ファイルを各自で書き換え

る必要がある場合もあります。

実用上最低限のシステムはフロッピーディスク2枚分のファイルに集約されます。フロッピーディスクの内容をそのままハードディスクに落とせば最低限のシステムができあがります。ただし、前述のとおり、FDベースで使用するようには設計されていませんので、必ずハードディスクにインストールして使用するようにしてください。

なお、ハードディスクに各種ファイルを転送し終えたあとに書き換える必要があるファイルは以下のとおりです。

AUTOEXEC.BAT（EFFECTとEX-SYSTEMにパスを通す）
¥EX-SYSTEM¥EX_STAMP.SYS
¥EX-SYSTEM¥EX_FONT.SYS（各自の環境にあわせる）
¥EFFECT¥PATTERNBRUSH.SYS
¥EFFECT¥PICPAINT.SYS

これらを各自の環境にあわせて設定してください。

続いて、¥EX-SYSTEM¥EXCONFIG.SYSをコピーしてEXWIN.SYS、ZS_EX.SYS、MAT_EX.SYSの3つのファイルを作成しておいてください。これでコマンドラインから、

A)EX_WIN
A)ZS_EX
A)MAT_EX

の入力で各ツールが起動するようになります（Z'sSTAFF、MATIERは別途必要）。

●その他のファイル

せっかくのCD-ROMですので、EX-System以外にもさまざまなファイルを入れてあります。詳しくはそれぞれのドキュメントなどをお読みください。また、一部プログラムのバージョンアップなどを目的としたものについては最低限のドキュメントしかありません。ご了承ください。

以下に主なファイルを挙げておきます。

・DoGA CGAシステム用背景データ集
DoGA CGAシステムで使用するための背景データ集です。柳沢PICで収録されていますので、EX-Systemでも簡単に使用できます。

・SION IV完成版（要4Mバイト）

MIDI対応になった

・Z-MUSICシステムver.2.07/ver.3 試用版

Z-MUSICシステムのPCMファイルなどを含んだ最新版実行ファイルなどをまとめました。

ver.3についてはまだまだデバッグ中のものですから使用には十分注意してください。

●プログラム&データについて

収録されたPhotoCD画像は小川俊毅氏、文月涼氏の提供によるものです。サンプルグラフィックは川原由唯氏、高橋哲史氏、森山知己氏、闇雲氏の提供によるものです。

ThinLine.XIは安森然氏の作品です。そのほか、収録されているDoGA CGAシステムはプロジェクトチームDoGA、3D Atrier体験版は株式会社マイクロネットの提供によるものです。

各種ツール、データを提供していただきました関係者の皆様にこの場を借りてお礼を申し上げます。

EX-System

Oh!X編集部編

プログラム作成

発行者

発行所

表紙CG

印刷所

菊地 功

橋本五郎

ソフトバンク株式会社 出版事業部

〒103 東京都中央区日本橋浜町3-42-3

販売 03(5642)8101

編集 03(5642)8122

森山知己

文唱堂印刷株式会社

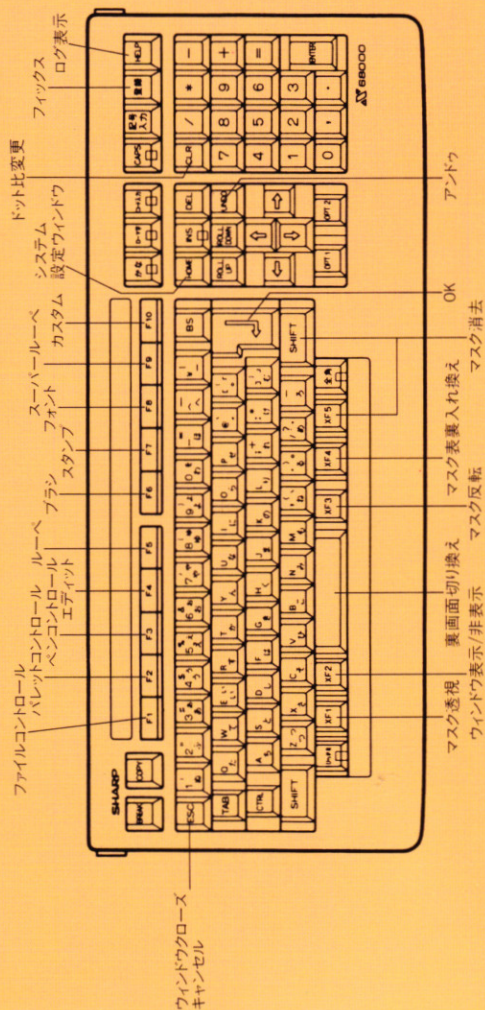
1996年 2 月29日

初版第 1 刷発行

Printed in Japan 1996.

落丁、乱丁本は小社販売局にてお取り替えいたします。
定価は裏表紙に表示してあります。

ISBN4-89052-879-2 C0055



ISBN4-89052-879-2

C0055 P4800E



9784890528790

定価4,800円
(本体4,660円)



1910055048003

EX-System愛読者プレゼント



オリジナルマグカップ

弊社ソフトバンクの製品をお買い上げいただきありがとうございます。今後(?)の編集の資料にいたしますので、付属ハガキのアンケートにお答えください。ご協力いただいた方のなかから抽選で以下のプレゼントをお送りいたします。

・オリジナルTシャツ

「桜」、「蟹」などの川原由唯氏のCG画像をTシャツにしたもの。各絵柄2枚、計8名の方に。

・オリジナルマグカップ

森山氏のCG画像をもとに菊地氏自らオープンで焼きあげた手作りの逸品。限定生産品を4名の方に。

なお、プレゼントの締め切りは3月31日到着分まで、プレゼント当選者の発表は発送をもって替えさせていただきます。

●サポートについて

Oh!X編集部が解散する都合上、雑誌上でのサポートなどは行うことができません。致命的なバグはないと思いますが、万一の場合は電腦倶楽部などをお願いすることになると思います。今後はデバッグ情報やサポートツールなどがソフトバンク発行の雑誌の付録CD-ROMにこっそり入っていたりするかもしれませんのでご注意ください。

郵便はがき

料金受取人払

日本橋局承認

673

差出有効期間
平成9年3月
17日まで

1 0 3 - 0 0

1 6 1

(受取人)

東京都中央区
日本橋浜町3-42-3

ソフトバンク株式会社

Oh!X編集部行

電話

住所

フリガナ

氏名

年齢

職業・勤務先
学校・学部・学年

所有機種

メモリ

CD-ROM ある・ない

EX-System



Tシャツ(SAKURA)



Tシャツ(PORTRAIT)

Tシャツのデザインは指定できません。サイズはM付のみとなっています。また,PORTRAITの絵柄は写真のものよりも大きめに変更されています。具体的な絵柄についてはCD-ROM内のPRESETディレクトリの中をごらんください。

EX-System

■この本を何でお知りになりましたか？

1. 雑誌広告 (雑誌名 /)
2. 雑誌の紹介記事で (雑誌名 /)
3. 書店で見て 4. 書店ですすめられて 5. 人にすすめられて
6. その他 ()

■この本をお買い上げの書店名は？

都・道
府・県

区・市

書店

■以下の質問にお答え下さい。

- | | | | |
|-----|-----------|--------|-----------|
| 内 容 | 1. わかりやすい | 2. ふつう | 3. わかりにくい |
| 装 丁 | 1. よい | 2. ふつう | 3. わるい |
| 価 格 | 1. 高い | 2. ふつう | 3. 安い |

■お読みになった感想をお聞かせ下さい。

■弊社発行の書籍・雑誌をお読みになったことがありますか？

書籍名 () 雑誌名 ()

■今後どのような企画をお望みですか？



Tシャツ(KANI2)



Tシャツ(SIRT07)

■プレゼント希望 (Tシャツ, マグカップ)

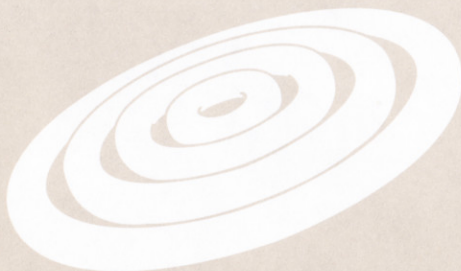
Graphic tool for X680x0

EX-System付録ディスク

ソフトバンク出版事業部
SOFTBANK CORP. 1996
MAXELL FLOPPY DISK

disk 1

maxell



100% Certified and Tested
100% Certifiées et Testées
Verificados y certificados al 100%

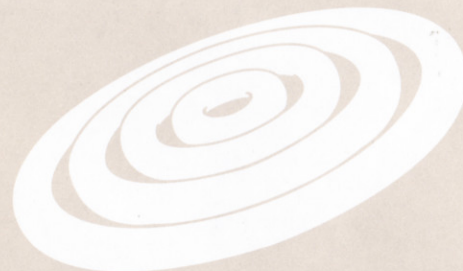
Graphic tool for X680x0

EX-System付録ディスク

ソフトバンク出版事業部
SOFTBANK CORP. 1996
MAXELL FLOPPY DISK

disk 2

maxell



100% Certified and Tested
100% Certifiées et Testées
Verificados y certificados al 100%

X68000

OSR-2BK-OH-MUR

EX-System

This is the Graphic tool for X680x0 series.
Required Human68K ver.3.0+CD-ROM driver,
4M bytes RAM&2~40M bytes harddisk space.

(c)1996 SOFTBANK CORP.