ICOT Technical Memorandum: TM-0785

TM-0785

PSI図形ツールのインストール方法

東　吉郎(JIPDEC)

August, 1989

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

ＰＳＩ図形ツールの
インストール方法

ＰＳＩ図形ツールのインストール方法

（1）インストール方法

①ライブラリアンを生成する。

②フロッピーをユニット0に入れる。

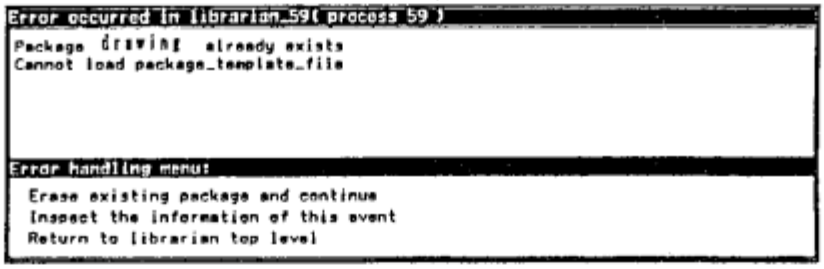
③ライブラリアンの「Ｅｘｅｃｕｔｅ」を選択する。

④Command>fdd(" drawing.com ") と入力後、リターンキーを入力する。

⑤FDD mount ok?
Input unit No. (0) :0
ユニット番号「0」を入力する（リターンキーは不要）。

⑥約1分かけてインストールされる。

◆既にパッケージｄｒａｗｉｎｇが存在していると、次のエラーメッセージが表示される。処理を続行する（既存のパッケージｄｒａｗｉｎｇが削除され、新しく生成される）か、中止かを選択する。



・処理を続行する場合（新バージョンをインストールする場合）

①「Erase existing package and continue 」を選択する。

②続行するかどうかの確認画面が、もう一度表示される。

③続行する場合には「ＥＲＡＳＥ」を、中止したい場合には「ＳＴＯＰ」を選択する。

④「ＥＲＡＳＥ」を選択すると、約1分かけてインストールされる。

・処理を中止する場合

①「Return to librarian top level 」を選択する。

②ライブラリアンの画面に戻る。

ＰＳＩ図形ツール
説明書

ＰＳＩ図形ツール　説明書　目次

1．概要

ＰＳＩ図形ツールは、マウスやキーボードを用いて対話的に、図形またはイメージを編集するシステムである。図形とは、円、矩形、線等から構成される図である。また、イメージとは、ウインドウ上のドットから構成されるデータである。

本ツールは図形処理プログラムとイメージ処理プログラムから構成される。図形処理プログラムは、任意のサイズのウインドウ上で図形を描画、編集し、イメージデータまたは図形データを生成する機能を提供する。イメージ処理プログラムは、任意のサイズのウインドウ上でイメージを描画、編集し、イメージデータを生成する機能を提供する。生成した図形データ、イメージデータは本ツール上で再編集が可能である。さらに、図形データはイメージ処理プログラムでイメージデータとして使用することもできる。

また、図形描画プログラム作成ユーティリティにより、図形データからその図形を描画するためのＥＳＰプログラムを生成することができる。

図形処理とイメージ処理の違いを簡単に説明する。

図形処理は、描画した図形を図形データとして管理する。図形データとは、描画図形の位置、大きさ、描画属性等を保持したデータである。図形処理において編集操作を行う場合、図形データをもとに図形単位で移動、消去等の操作が可能である。

一方、イメージ処理では、描画した図形を図形データとして管理せずに単なるビットマップ・イメージとして扱う。イメージ処理において編集操作を行う場合、領域単位のイメージとして移動、消去、拡大・縮小、回転等の操作が可能である。

2．操作

2．1　起動方法

(1)　メソッド呼び出しによる起動

```
:drawing(#drawing##psi__drawing __tool)
```

メソッドを呼び出したプロセス上で実行する。

(2)　システム・メニューからの起動

ログインファイルのmenu項目のitems __listにpsi __drawing __toolを追加する。

例）

```
menu:-
        items __list([
                        {'debugger', debugger},
                                  •
                        {'PsiDraw', drawing##psi __drawing __tool}]).
```

２．２　初期操作

本システムを起動すると、図２－１に示す処理選択メニューが表示される。
'drawing' を選択すると図形処理プログラム、'image' を選択するとイメージ処理プログラムが起動される。

図２－１　処理選択メニュー

'drawing' 、'image' のいずれかを選択すると図２－２に示すＰＳＩ図形ツール・ウインドウが表示される。

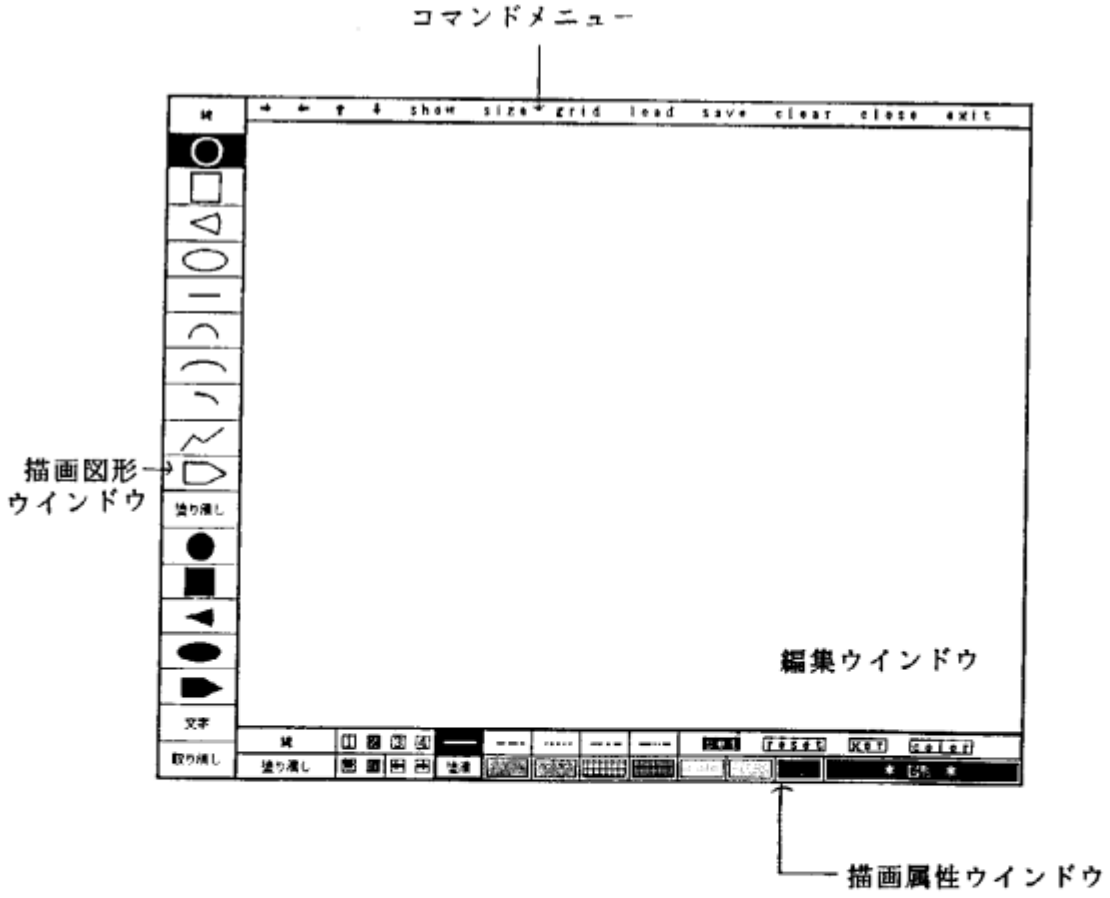


図２－２　ＰＳＩ図形ツール・ウインドウ

ＰＳＩ図形ツール・ウインドウは、図形処理プログラム、イメージ処理プログラムに共通のものであり、ウインドウの右下にどちらのプログラムが起動されているかを表示している。このウインドウ上で図形描画、編集操作を行う。

２．３　基本操作・機能

本ツールの操作および機能の概略を説明する。

ＰＳＩ図形ツール・ウインドウは、編集ウインドウ、コマンドメニュー、描画図形ウインドウ、描画属性ウインドウの４種のウインドウから構成される（図２－２）。編集ウインドウは、図形を描画、編集するウインドウであり、サイズ変更、グリッド表示、スクロール等の機能を持っている（詳細は、２．４　編集ウインドウ操作を参照）。コマンドメニューは、プログラムの終了、データのセーブ／ロードおよび編集ウインドウの状態（サイズ、表示位置）等を変更するためのコマンドを表示しているウインドウである。描画図形ウインドウは、描画図形を選択するウインドウである。描画属性ウインドウは、図形を描画する際に必要な線種、線幅、塗り潰しパターン、塗り潰しパターンの外枠の有無、描画時の論理演算などを選択するウインドウである。編集ウインドウ上で描画操作を行う場合、描画図形ウインドウ、描画属性ウインドウで反転表示されている図形と属性により描画を行う。図形や属性の選択は、選択したい図形または属性が表示されている位置でのマウスクリックにより行う。詳細は、２．５　描画図形選択操作を参照のこと。

編集ウインドウ上でマウス操作をすることにより、図形を描画、編集することができる。編集ウインドウでのマウス操作を簡単に説明する。

左クリック・・・図形を描画する。詳細は、２．６　描画操作を参照。

中クリック・・・図形またはイメージを編集をする。図形処理とイメージ処理で操作および機能が異なる。詳細は、２．７　編集操作を参照。

右クリック・・・描画、編集操作を中止する。

また、コマンドメニューの項目を選択することにより以下のことができる。

(1) →

編集ウインドウを右にスクロールする。

(2) ←

編集ウインドウを左にスクロールする。

(3) ↓

編集ウインドウを下にスクロールする。

(4) ↑

編集ウインドウを上にスクロールする。

(5) show

編集ウインドウの全イメージを1/3 に縮小したウインドウを表示する。このウインドウ上でマウス操作を行うことにより編集ウインドウの表示位置を変更できる。詳細は、２．４　編集ウインドウ操作を参照。

(6) grid

編集ウインドウにグリッド（格子）を表示する。グリッドは、編集ウインドウでのマウス操作を容易にするためのものである。詳細は、２．４　編集ウインドウ操作を参照。

(7) size

図形描画、編集領域のサイズを変更する。サイズ指定のない場合、既定値はスクリーンサイズである。最大サイズは、2000×2000ドットである。詳細は、２．４　編集ウインドウ操作を参照。

(8) load

既存のデータファイルを再編集する。詳細は、２．９　データファイル再編集を参照。

(9) save

現在編集ウインドウに描画されている状態を、データファイルとして固定ディスク上に格納する。詳細は、２．８　データファイル作成を参照。

(10) clear

編集ウインドウに描画されている図形またはイメージを消去する。

(11) close

ＰＳＩ図形ツール・ウインドウを消去しアイコンを表示する。

(12) exit

本プログラムを終了する。

２．４　編集ウインドウ操作

編集ウインドウのサイズ変更、スクロール機能、グリッド表示について説明する。

(1) サイズ変更

図形を描画、編集できる大きさ（描画編集サイズ）は、最大2000×2000ドットである。描画編集サイズが編集ウインドウより大きい場合、編集ウインドウは、その一部を表示し、小さい場合は編集ウインドウがその大きさになる。描画編集サイズの変更はコマンドメニューの "size" により行う。 "size" を選択すると、図２－３に示すサイズ設定ウインドウが表示される。サイズ設定ウインドウに表示されている値は、現在設定されている描画編集サイズの値であり、その値を変更すること

により描画編集サイズを変更することができる。

変更した描画編集サイズが編集ウインドウの既定サイズ（起動時のサイズ）より小さい場合、描画編集サイズの矩形枠がＰＳＩ図形ツール・ウインドウ上に表示されるので、適当な位置でマウスクリックすると、編集ウインドウの表示位置が設定される。

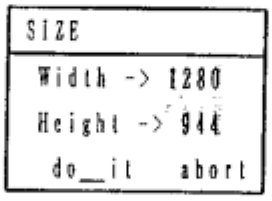


図２－３　サイズ設定ウインドウ

(2) スクロール機能

描画編集サイズが編集ウインドウの既定サイズより大きい場合、編集ウインドウは、スクロール機能を持つ。スクロールは、コマンドメニューの "→" 、 "←" 、 "↑" 、 "↓" または "show" により行う。 "→" 、 "←" 、 "↑" 、 "↓" を選択すると、編集ウインドウは、少しずつ矢印の方向へスクロールする。また、 "show" を選択すると、描画編集サイズを1/3 に縮小した全イメージを示したウインドウを表示する。このウインドウ上でマウスを左一回クリックすると、現在表示している領域を矩形枠で示す。もう一度マウスを左一回クリックすると矩形枠が自由に動くので、適当な位置でマウスクリックすると、その領域を編集ウインドウ上に表示する。

(3) グリッド表示

描画、編集操作を容易にするために、編集ウインドウ上にグリッドを表示することができる。グリッド表示は、コマンドメニューの "grid" により行う。 "grid" を選択すると、図２－４に示すグリッド設定ウインドウが表示される。グリッド設定ウインドウで、グリッド表示の有無（grid）、グリッドの大きさ（size）、マウス入力単位（unit）を設定することができる。表示されている値は、現在設定されている値である。適当な値を設定することにより、グリッド表示の有無、グリッドの大きさ、マウス入力単位を変更することができる。なお、グリッドの単位はドット単位であり、１から９９までの整数が入力可能である。また、マウス入力単位unitの設定を "grid" に設定すると、編集ウインドウ上でマウス入力を行った際、グリッドの隣接する４格子点の左上隅の点で入力があったとみなす。

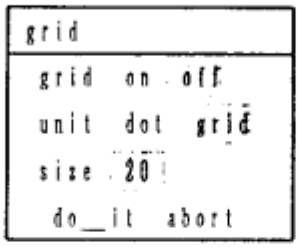


図２－４　グリッド設定ウインドウ

２．５　描画図形選択操作

編集ウインドウ上での図形の描画は、描画図形ウインドウ、描画属性ウインドウの反転表示の組み合わせで行う。描画図形ウインドウは、描画図形の種類を選択するウインドウであり、描画属性ウインドウは、線種、線幅、塗り潰しパターン、塗り潰しパターンの外枠の有無、描画時の論理演算などを選択するウインドウである。図２－５に描画図形ウインドウ、図２－６に描画属性ウインドウを示す。

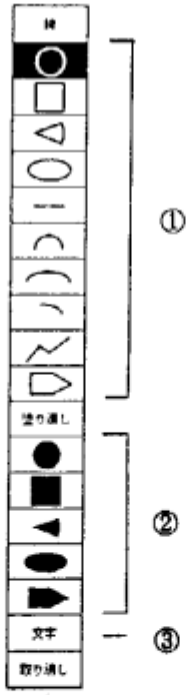


図２－５　描画図形ウインドウ

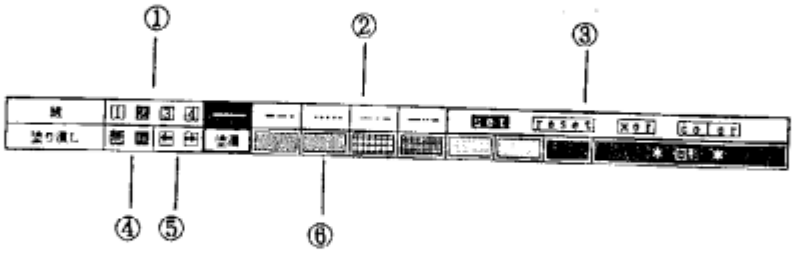


図２－６　描画属性ウインドウ

描画図形ウインドウで選択できる図形は、図２－５に示すように①線図形、②塗り潰し図形、③文字の３種に分類される。

線図形の場合の描画属性は、描画属性ウインドウの線種（図２－６の②）、線幅（図２－６の①）の項目で指定する。

塗り潰し図形の場合の描画属性は、描画属性ウインドウの塗り潰しパターンの外枠の有無（図２－６の④）と塗り潰しパターン（図２－６の⑥）の項目で指定する。なお、図２－６の⑤に表示されている矢印をマウスクリックすることにより、塗り潰しパターンの次候補を表示することができる。

また、図形描画時の論理演算は、図２－６の③の項目で指定する。論理演算として、set, reset, xor (exclusive or) とcolor （色指定）がある。color を指定すると図２－７に示すカラー設定ウインドウが表示される。

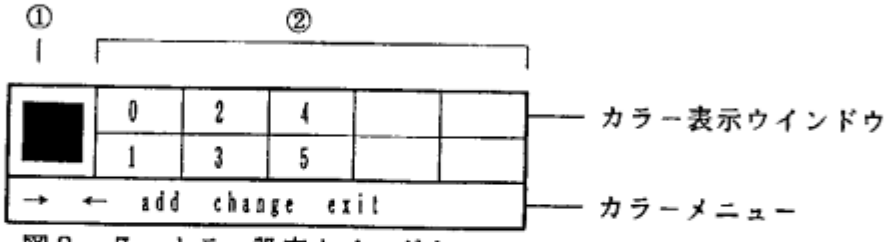


図２－７　カラー設定ウインドウ

カラー設定ウインドウは、カラー表示ウインドウとカラーメニューから成る。さらに、カラー表示ウインドウは、①設定カラー表示部、②登録カラー表示部から成る。登録カラー表示部は、現在登録されているカラー番号とその色を表示している。設定カラー表示部は、現在描画カラーとして設定されている色を表示している。登録カラー表示部でマウスクリックすると、その色に描画カラーを変更することができる。カラー設定ウインドウで一度に表示できる色は１０色であり、カラー番号１０番以降を表示したい場合は、カラーメニューの"→"をマウスクリックする。カラーメニューの"→"、"←"は、カラー表示ウインドウに表示する色を変更するために用いる。登録している色が１０色以下の時、この操作は無効である。カラーメニューの"add"を選択するとカラー・ボードが表示され、自由に色の追加を行うことができる。カラー・ボードは、操作説明書（Ⅱ）Ⅰ－２．２．４カラー・ボードを参照。また、カラーメニューの"change"を選択すると、変更番号を入力するためのウインドウが表示される。

正しい番号を入力すると、その番号と番号の色を示したカラー・ボードが表示される。カラー・ボード上で適当な色を作成するとその色に変更することができる。カラーメニューの"exit"を選択すると色指定操作が終了する。

２．６　描画操作

描画図形ウインドウの個々の図形の意味、編集ウインドウ上での描画方法を説明する。

ただし、描画図形ウインドウ上の"取消"は、編集ウインドウを一つ前の描画状態に戻すために用いるものである。また、描画中の操作中止は、右一回マウスクリックすることにより行う。


・・・　円を描く。初めに左一回マウスクリックした位置を中心とし、次に左一回マウスクリックした位置までを半径とする円を描く。


・・・　矩形を描く。初めに左一回マウスクリックした位置を左上頂点とし、次に左一回マウスクリックした位置を右下頂点とする矩形を描く。


・・・　扇形を描く。初めに左一回マウスクリックした位置を中心とし、次に左一回マウスクリックした位置を円弧開始点、さらに左一回マウスクリックした位置を円弧終了点とする扇形を描く。なお、扇形の弧は時計と逆回りに描画する。

 ・・・ 楕円を描く。初めに左一回マウスクリックした位置を左上頂点とし、次に左一回マウスクリックした位置を右下頂点とする矩形枠に内接した楕円を描く。

 ・・・ 直線を描く。初めに左一回マウスクリックした位置を開始点、次に左一回マウスクリックした位置を終了点とする直線を描く。

 ・・・ 円弧を描く。初めに左一回マウスクリックした位置を中心とし、次に左一回マウスクリックした位置を円弧開始点、さらに左一回マウスクリックした位置を円弧終了点とする円弧を描く。なお、円弧は時計と逆回りに描画する。

 ・・・ 半楕円弧を描く。初めに左一回マウスクリックした位置を左上頂点とし、次に左一回マウスクリックした位置を右下頂点とする矩形枠に接する半楕円弧を描く。なお、描画図形ウインドウでこの図形を選択する際、マウスを左二回クリックすると、図２－８に示す半楕円弧の種別を決定するメニューが表示される。初期設定は、上半楕円であり、マウスクリックにより変更することができる。

楕円描画方法
上半楕円
下半楕円
右半楕円
左半楕円

図２－８　半楕円描画メニュー

 ・・・ 1/4 楕円弧を描く。初めに左一回マウスクリックした位置を左上頂点とし、次に左一回マウスクリックした位置を右下頂点とする矩形枠に接する1/4 楕円弧を描く。なお、描画図形ウインドウでこの図形を選択する際、マウスを左二回クリックすると、図２－９に示す1/4 楕円弧の種別を決定するメニューが表示される。初期設定は、右上1/4 楕円であり、マウスクリックにより変更することができる。

楕円描画方法
右上1/4 楕円
右下1/4 楕円
左上1/4 楕円
左下1/4 楕円

図２－９　1/4 楕円描画メニュー

 ・・・　折れ線を描く。初めに左一回マウスクリックした位置を始点とし、左一回マウスクリックした点を順につないだ折れ線を描く。左二回マウスクリックすると、その点を終点として操作を終了する。なお、折れ線描画操作中の中一回クリックは、一つ前のクリックをキャンセルする。

 ・・・　多角形を描く。初めに左一回マウスクリックした位置を始点とし、続いて左一回マウスクリックした点を順につないでいき、左二回マウスクリックした点を終点とする多角形を描画する。なお、描画操作中の中一回クリックは、一つ前のクリックをキャンセルする。

 ・・・　塗り潰した円を描く。初めに左一回マウスクリックした位置を中心とし、次に左一回マウスクリックした位置までを半径とする塗り潰した円を描く。

 ・・・　塗り潰した矩形を描く。初めに左一回マウスクリックした位置を左上頂点とし、次に左一回マウスクリックした位置を右下頂点とする塗り潰した矩形を描く。

 ・・・　塗り潰した扇形を描く。初めに左一回マウスクリックした位置を中心とし、次に左一回マウスクリックした位置を円弧開始点、さらに左一回マウスクリックした位置を円弧終了点とする塗り潰した扇形を描く。なお、扇形の弧は時計と逆回りに描画する。

 ・・・　塗り潰した楕円を描く。初めに左一回マウスクリックした位置を左上頂点とし、次に左一回マウスクリックした位置を右下頂

点とする矩形枠に内接した塗り潰した楕円を描く。

 ・・・ 塗り潰した多角形を描く。初めに左一回マウスクリックした位置を始点とし、続いて左一回マウスクリックした点を順につないでいき、左二回マウスクリックした点を終点とする塗り潰した多角形を描画する。なお、描画操作中の中一回クリックは、一つ前のクリックをキャンセルする。

文字 ・・・ 文字列を描く。文字列描画には、サイズ指定と矩形指定の２通りの方法がある。サイズ指定は、一文字の縦、横のサイズを指定し、その大きさで文字を描画する方法である。矩形指定は、指定された矩形領域に文字を拡大・縮小して描画する方法である。

サイズ指定の場合は、左一回マウスクリックで出力する文字列の左上隅の位置を設定する。続いて文字列入力ウインドウが表示されるので、文字列を入力する。文字列入力ウインドウ上でリターンキーを入力すると文字列入力ウインドウは消え、指定した左上隅の位置から文字列を出力する。

矩形指定の場合は、左一回マウスクリックを左上隅、次の左一回マウスクリック右下隅とする矩形領域を設定する。続いて文字列入力ウインドウが表示されるので、文字列を入力する。文字列入力ウインドウ上でリターンキーを入力すると文字列入力ウインドウは消え、指定した矩形領域に文字列を出力する。

なお、描画図形ウインドウ上で“文字”を選択する際、左二回マウスクリックすると、図２－１０に示す文字列描画属性ウインドウが表示される。文字列描画属性ウインドウ上で、フォント、描画指定（サイズ指定、矩形指定）、形式（縦書き、横書き）を指定することができる。サイズ指定の場合は、一文字の縦、横の大きさ（ドット単位）を指定する必要がある。また、最大文字数を指定すると最大文字数以降は、次行（横書きの場合）または次列（縦書きの場合）に折返して文字を出力する。なお、最大文字数を指定しない場合は“＊”を入力する。

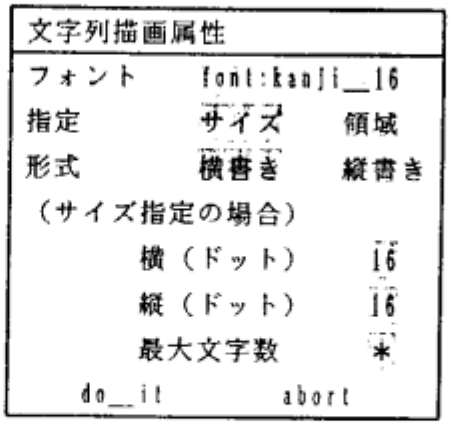


図２－１０　文字列描画属性ウインドウ

２．７　編集操作

編集ウインドウ上に描画されている図形またはイメージの編集方法を説明する。編集方法は、図形処理、イメージ処理により異なるため、個々に説明する。ただし、編集中の右一回マウスクリックは、図形処理、イメージ処理共に編集操作を中止するためのものである。

(1) 図形処理編集操作

操作を説明する前に、図形処理における図形管理について説明する。

図形処理では、図形データをもとに図形単位で移動、消去等の編集操作を行うため、描画した図形を矩形で管理する。例えば、直線を描画した場合、直線は図２－１１の点線で示した矩形で管理される。

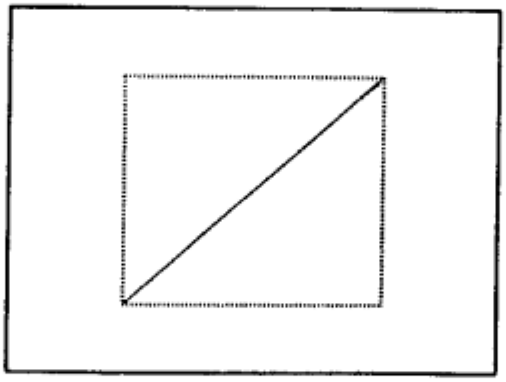


図２－１１　図形管理例

以後、図形処理編集操作の説明で“選択されている図形”とは、図２－１１の点線で囲まれた矩形内に描画されている図形をさす。

編集ウインドウ上で、中一回マウスクリックすると、編集図形を選択することができる。図形が選択されると、その図形を管理している矩形枠を点滅表示する。図形が重なりあっている場合、さらに中一回マウスクリックすることによって別の図形を選択することができる。

図形が選択されている状態で中二回マウスクリックすると、図２－１２に示す図形処理編集メニューが表示される。

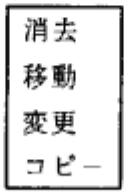


図２－１２　図形処理編集メニュー

また、図形が選択されている状態であっても左一回マウスクリックすると、描画属性ウインドウの線種・線幅または塗り潰しパターン・塗り潰しパターンの外枠の指定状態を変更することができる。

図形処理編集メニューの各項目が選択された場合の操作を示す。

消去　・・・　選択されている図形を消去する。

移動　・・・　選択されている図形を移動する。“移動”を選択するとマウスが点線枠と同じ大きさの矩形に変化する。適当な位置でマウスクリックすると図形が移動する。

変更　・・・　選択されている図形の線種・線幅または塗り潰しパターン・塗り潰しパターンの外枠を、現在描画属性ウインドウ上で指定されている状態に変更する。

コピー・・・　選択されている図形と同じ図形を描画する。“コピー”を選択するとマウスマーカが点線枠と同じ大きさの矩形に変化する。適当な位置でマウスクリックすると、その位置に図形がコピーされる。

(2) イメージ処理編集操作

編集ウインドウ上で中クリック（中一回、中二回どちらでもよい）すると、図２－１３に示すイメージ処理編集メニューが表示される。

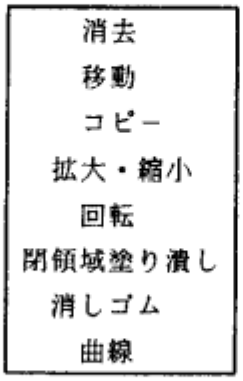


図２－１３　イメージ処理編集メニュー

イメージ処理編集メニューの各項目が選択された場合の操作を説明する。

消去・・・・・・　矩形内のイメージを消去する。初めに左一回マウスクリックした位置から次に左一回マウスクリックした位置までの矩形内のイメージを消去する。

移動・・・・・・　矩形内のイメージを移動する。初めに左一回マウスクリックした位置から次に左一回マウスクリックした位置までの矩形内のイメージを、次にマウスクリックした位置が矩形の左上頂点となる位置に移動する。なお、左二回マウスクリックで“移動”を選択すると、転送の論理演算を変更するメニューが表示され、転送の論理演算を変更することができる。転送の論理演算は、exclusive _or（移動位置にある元のイメージとの反転）、copy（背景色）、reverse（背景色の反転）のいずれかであり、既定値は、copyである。

コピー・・・・・・　矩形内のイメージをコピーする。初めに左一回マウスクリックした位置から次に左一回マウスクリックした位置までの矩形内のイメージを、次にマウスクリックした位置が矩形の左上頂点となる位置にコピーする。なお、左二回マウスクリックで“コピー”を選択すると、転送の論理演算を

変更するメニューが表示され転送の論理演算を変更することができる。転送の論理演算は、exclusive _or（コピー位置にある元のイメージとの反転）、copy（背景色）、reverse（背景色の反転）のいずれかであり、既定値は、copyである。

拡大・縮小・・・　矩形内のイメージを拡大・縮小する。初めに左一回マウスクリックした位置から次に左一回マウスクリックした位置までの矩形内のイメージを、次に左一回マウスクリックした大きさに拡大または縮小する。なお、左二回マウスクリックで"拡大・縮小"を選択すると、拡大・縮小のタイプを変更するメニューが表示され、拡大・縮小のタイプを変更することができる。拡大・縮小のタイプは、ウインドウ・システムで提供しているmagnify とexpandのいずれかであり、既定値は、expandである。

回転・・・・・・　矩形内のイメージを回転する。初めに左一回マウスクリックした位置から次に左一回マウスクリックした位置までの矩形内のイメージを回転する。矩形を指定すると、矩形の左下頂点からマウスの軌跡と共に直線を表示する。適当な位置でマウスクリックすると、矩形の左下からマウスクリックした直線までの角度（時計と逆回りに角度をとる）を回転角とする回転を行う。

閉領域塗り潰し・　閉領域を塗り潰す。左一回マウスクリックした点を含む閉領域を描画属性ウインドウで選択されている塗り潰しパターンで塗り潰す。なお、左二回マウスクリックで"閉領域塗り潰し"を選択すると、特定の矩形領域内をクリッピングして閉領域の塗り潰しを行うことができる。その場合、矩形を指定（左一回マウスクリックした位置から次に左一回マウスクリックした位置までの矩形）した後、指定した矩形内で左一回マウスクリックをする。

消しゴム・・・・　イメージを消去する。左一回マウスクリックで選択するとマウスが消しゴムに変化する。この状態で左マウスを押し続けると、その部分のイメージを消去する。この操作を中

止したい場合は、右一回マウスクリックする。なお、左二回マウスクリックで“消しゴム”を選択すると、消しゴムの太さを変更するメニューが表示され、太さを変更することができる。太さは、1(1*1 ドット) から8(8*8 ドット) のいずれかであり、既定値は5 である。

曲線　・・・・　曲線を描画する。左一回マウスクリックするとマウスが鉛筆に変化する。この状態で左マウスを押し続けると、マウスの軌跡にしたがって曲線を描画する。この操作を中止する場合は、右一回マウスクリックする。なお、左二回マウスクリックで“曲線”を選択すると、曲線の太さを変更するメニューが表示され、太さを変更することができる。太さは、1(1*1 ドット) から8(8*8 ドット) のいずれかであり、既定値は5 である。

2．8　データファイル作成

本ツールは、編集ウインドウ上で描画、編集した図形またはイメージを、データファイル（図形データファイルまたはイメージデータファイル）として固定ディスクに格納することができる。本ツールから直にファイルを格納できるのは固定ディスクのみであり、直接FDD に格納することはできない。FDD に格納する場合は、一度固定ディスクに格納した後、ファイル・マニピュレータを介して行う必要がある。なお、FDD に格納してあるデータファイルをファイル・マニピュレータから固定ディスクにコピーする場合、図形データファイルは、'copy __test' 、イメージデータファイルは、'copy'により行わなければならない。

図形データファイルは、ファイル名が、'***.draw'のものであり、図形処理プログラム上で再編集を行えるほか、イメージ処理プログラム上にイメージデータとしてロードすることもできる。また、図形描画プログラム作成ユーティリティにより、図形を描画するためのＥＳＰプログラムに変換することができる。

イメージデータファイルは、ファイル名が、'***.image' のものであり、イメージ処理プログラム上で再編集を行えるほか、ウインドウシステムが提供しているメソッドにより、ウインドウ上にイメージを描画することができる。

図形処理プログラムを起動している場合、図形データファイルとイメージデータファイルを作成することができる。イメージ処理プログラムを起動している場合は、イメージデータファイルのみ作成できる。ファイル作成操作はコマンドメニューの

'save'を選択することにより行う。'save'を選択した後の操作方法を説明する。

(1) 図形処理プログラム

'save'を選択すると、図２－１４に示すファイルの種類を決めるセーブメニューが表示される。

図２－１４　セーブメニュー

各項目が選択された場合の操作方法を説明する。

draw　・・・　図形データファイルを作成する。'draw'を選択するとファイル名入力ウインドウが表示される。キーボードからファイル名またはファイル・パス名を入力する。ファイル名には、拡張子'draw'を付ける。ファイル識別子が無い場合、システムがファイル拡張子'draw'を付ける。正しいファイル名を入力すると、図形データファイルが作成される。

image　・・・　イメージデータファイルを作成する。'image' を選択すると図２－１５に示すイメージセーブ領域指定ウインドウが表示される。

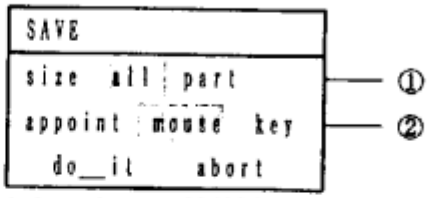


図２－１５　イメージセーブ領域指定ウインドウ

イメージセーブ領域指定ウインドウは、セーブするイメージの位置と大きさを指定するウインドウである。①部で'all' を選択すると、セーブされるイメージの大きさは、描画編集サイズ（現在図形を描画、編集している大きさ）となる。'part'を選択すると、ある特定部分のイメージを抜き出してイメージデータとしてファイルにセーブすることができる。その場合、②部

でイメージを抜き出す操作をマウス入力(mouse) にするか、キーボード入力(key) にするか指定しなければならない。マウス入力を指定すると、マウスマーカが "┏" に変わるので、マウスマーカによりファイルにセーブするイメージ領域を決める。キーボード入力を指定すると、図２－１６に示すイメージ領域指定ウインドウが表示されるので左上端の座標とサイズを入力してファイルにセーブするイメージ領域を決める。なお、イメージ領域指定ウインドウに表示されている値は、描画編集サイズである。

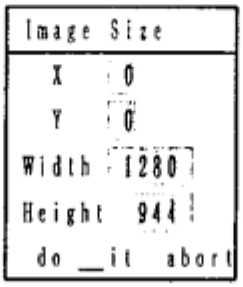


図２－１６　イメージ領域指定ウインドウ

セーブする領域が決まるとファイル名入力ウインドウが表示される。キーボードからファイル名またはファイル・パス名を入力する。ファイル名には、ファイル拡張子'image' を付ける。ファイル識別子が無い場合、システムがファイル拡張子'image' を付ける。正しいファイル名を入力すると、イメージデータファイルが作成される。

(2) イメージ処理プログラム

'save'を選択すると、イメージデータファイルを作成する。作成方法は、図形処理プログラムのセーブメニューで'image' を選択した場合と同じである。

２．９　データファイル再編集

本ツールでは、既存の図形データファイルまたはイメージデータファイルを再編集することができる。再編集操作はコマンドメニューの'load'を選択し、既存のデータファイルをロードした後に行う。

図形データファイルとは、ファイル拡張子が'draw'の図形処理プログラムで作成

したファイルである。

イメージデータファイルとは、ファイル識別子が'image' のイメージ処理プログラムで作成したファイル、あるいはウインドウシステムが提供しているメソッドによりユーザが独自に作成したイメージファイルである。ただし、ユーザが独自に作成したイメージファイルをロードする場合、ファイル識別子が'image' でなければならない。

図形処理プログラムを起動している場合、図形データファイルのみロードできる。イメージ処理プログラムを起動している場合は、イメージデータファイルまたは図形データファイルをイメージデータとしてロードすることができる。'load'を選択した後の操作方法を説明する。なお、描画編集中にロードを実行した場合、描画編集中の図形またはイメージは消去される。また、描画編集サイズは、ロードしたファイルに登録されている大きさとなる。

(1) 図形処理プログラム

'load'を選択すると、ファイル名入力ウインドウが表示される。キーボードからファイル名またはファイル・パス名を入力する。正しいファイル名（ファイル拡張子が'draw'）を入力すると、その図形データファイルがロードされる。

(2) イメージ処理プログラム

'load'を選択すると、図２－１７に示すファイルの種類を選ぶロードメニューが表示される。

図２－１７　ロードメニュー

'draw'を選択すると図形データファイル、'image' を選択するとイメージデータファイルをロードするためのファイル名入力ウインドウが表示される。図形データファイルをロードする場合は、ファイル拡張子が'draw'、イメージデータファイルをロードする場合は、ファイル拡張子が'image' のファイル名を入力するとそのデータファイルがロードされる。

3．図形描画プログラム作成ユーティリティ

ＰＳＩ図形ツールの図形処理プログラムを用いて生成した図形データから、その図形を描画するＥＳＰソース・プログラムを作成するためのユーティリティである。

本ユーティリティとＰＳＩ図形ツールを合わせて使用することにより、ウインドウに図形を描画するためのクラスを容易に作成することができる。

3．1　起動方法

（1）　メソッド呼び出しによる起動

```
:generate(#drawing##drawing __class __generator)
```

メソッドを呼び出したプロセス上で実行する。

（2）　システム・メニューからの起動

ログインファイルのmenu項目の items__listに drawing__class __generator をクラス名として項目を追加する。

例）

```
menu :-
        items __list([
                ['debugger', debugger],
                ['pmacs', pmacs],
                        .
                ['DrawGen', drawing##drawing __class __generator] ]
                ).
```

3．2　使用方法

本ユーティリティを起動すると図３－１に示す属性設定ウインドウが表示される。

適当な値を入力・設定した後、do__itを選択することにより図形描画プログラムを作成することができる。また、処理を中止したい場合はabort を選択する。

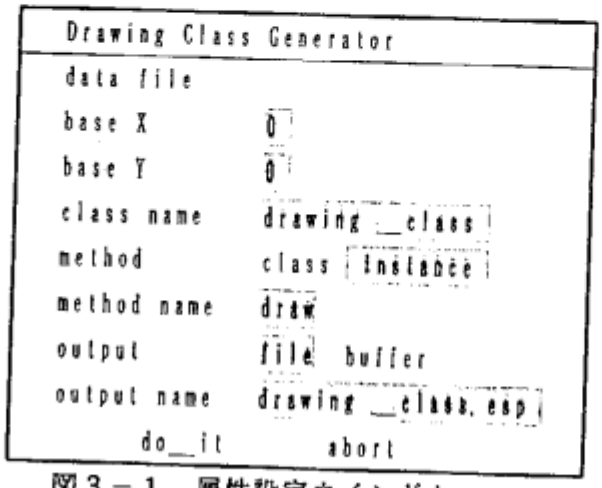


図３－１　属性設定ウインドウ

属性設定選択ウインドウの各項目について説明する。

（1）　d a t a　f i l e

図形データがセーブされているファイルのパス名。

なお，拡張子を指定しない場合は、'draw'が自動付加される。

指定したパス名のファイルが無い場合は、エラー・メッセージが表示された後、属性設定選択ウインドウが再表示される。

（2）　b a s e　X

作成された図形描画プログラムを実行するときに、各描画メソッドのＸ座標を表わす値に、本項目で設定した値を足し込んでメソッドを実行する。

図形全体をウインドウ内でＸ方向に平行移動したい場合に、本項目で移動量を調節する。

既定値は０。

（3） base Y

作成された図形描画プログラムを実行するときに、各描画メソッドのＹ座標を表わす値に、本項目で設定した値を足し込んでメソッドを実行する。

図形全体をウインドウ内でＹ方向に平行移動したい場合に、本項目で移動量を調節する。

既定値は０。

（4） class name

図形描画プログラムとして作成されるクラスのクラス名。

既定値は drawing_class 。

（5） method

作成された図形描画プログラムを起動するためのメソッドを、クラス・メソッドにするのかインスタンス・メソッドにするのかを指定する。

既定値はインスタンス・メソッド。

（6） method name

作成された図形描画プログラムを起動するためのメソッドのメソッド名。

（7） output

図形描画プログラムの出力先を指定する。

出力先として、ファイルとテキスト・バッファが指定できる。

既定値はファイル。

（8） output name

図形描画プログラムの出力先の名前を指定する。

出力先がファイルの場合はファイルのパス名、テキスト・バッファの場合はバッファ名である。

指定したパス名のファイルがすでにある場合は、バージョンを一つ増やしてファイルを生成する。

指定した名前のバッファがすでにある場合は、その旨警告メッセージを表示した後属性設定ウインドウを再表示する。

既定値は drawing_class 。

なお、出力先がファイルの場合は拡張子として'esp' が自動的に付加される。

3．3　図形描画プログラム

本ユーティリティにより作成される図形描画プログラムについて説明する。
図３－２に示すように設定した場合の図形描画プログラムの例を示す。

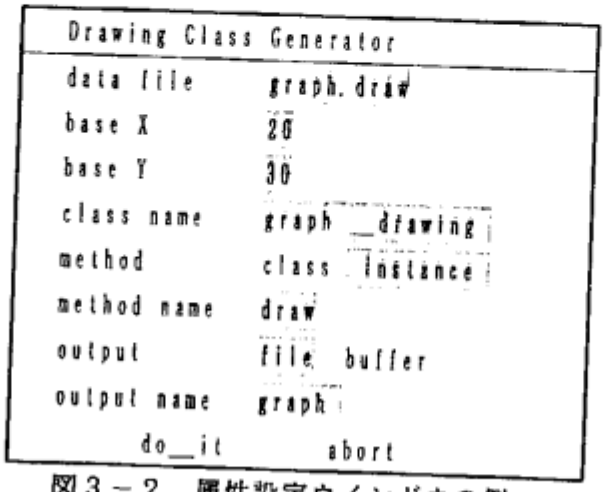


図３－２　属性設定ウインドウの例

（図形描画プログラムの例）

```
class  graph__drawing  has

instance

%  インスタンス属性スロット
  attribute
    base__x := 20,        %  スロット値だけ図形全体をＸ方向に移動する。
    base__y := 30,        %  スロット値だけ図形全体をＹ方向に移動する。
    font__bank            %  フォントオブジェクトを保持する。
    ;
```

```
%  インスタンス述語
%  :get__window/2
%        ウインドウの初期化を行う。
%        第２引数が未定義変数の場合は、ウインドウ・オブジェクトの生成及
%      びカラー設定を行い、オブジェクトを返す。
%        第２引数にウインドウ・オブジェクトを指定した場合は、カラー設定
%      のみ行う。描画時のOperation にカラー番号を指定している場合は、図
%      形描画メソッド(:draw/2) を呼ぶ前にカラー設定をしなければならない。

:get__window(Object, Window) :-
    get __window(Window)
    ;

%  :draw/2
%        ウインドウに図形を描画する。
%        第２引数は、ウインドウ・オブジェクトを指定する。
:draw(Object, Window) :-
    draw(Window, Object!base __x, Object!base __y, Object)
    ;

%  :reset__base/3
%        スロットbase__x 、base__y を変更する。
:reset__base(Object, X, Y) :-
    Object!base __x := X,
    Object!base __y := Y
    ;
```

```
local

%  ローカル述語
%  get __window/1
%        ウインドウの初期化を行う。
%        第１引数が未定義変数の場合は、ローカル述語window__class/1, init
%      iation__parameter/1 の情報に従ってウインドウを生成し、カラー設定
%      を行う。
%        第１引数にウインドウ・オブジェクトが指定されている場合は、その
%      ウインドウに対してカラー設定を行う。
get __window(Window) :-
    unbound(Window), !,
    window__class(Class),
    initiation__parameter(Init),
    :create(Class, Init, Window),
    set __color(Window)
    ;
get __window(Window) :-
    set __color(Window)
    ;

%  window__class/1
window__class(Class) :-
    Class = #standard __window
    ;

%  initiation__parameter/1
initiation__parameter(Init) :-
    Init = [
            size(1280,944)
          ]
    ;
```

```
%  set__color/1, set__color/2
%        ローカル述語 color__pairs __list/1の情報に従ってウインドウにカ
%     ラーを登録する。
set__color(Window) :-
    color__pairs__list(Color__pairs__list),
    set__color(Color__pairs__list, Window)
    ;
set__color([], Window) :- ! ;
set__color([[Even__number, Colors] | Color__pairs__list], Window) :-
    :set__color__pair(Window, Even__number, Colors),
    set__color(Color__pairs__list, Window)
    ;

%  color__pairs__list/1
color__pairs__list(Color__pairs__list) :-
    Color__pairs__list =
     [[0.0], [2, [[30720, 5376, 18944], [32543, 65535, 0]]]]
     ;
```

```
%  draw/4
%        ローカル述語data/2に記述されている図形描画情報のＸ座標及びＹ座
%     標の値にそれぞれのベース値を足し込み、後戻り機能を利用して図形描
%     画を行なう。
draw(Window, Base __x, Base __y, Object) :-
    data(Method, Args),
    modify(Method, Base __x, Base __y, Args, Arguments, Object),
    :refute(Window, Method, Arguments),
    fail
    ;
draw(Window, Base __x, Base __y, Object) ;

%  modify/6
%        各描画メソッド毎にＸ座標及びＹ座標の値にそれぞれのベース値を足
%     し込む。
modify(draw __line, Base__x, Base __y,
                        Args, Arguments, Object) :- !,
    stack __vector(Args, Length),
    new __stack __vector(Arguments, Length),
    adjust(Base __x, 0, Args, Arguments),
    adjust(Base __y, 1, Args, Arguments),
    adjust(Base __x, 2, Args, Arguments),
    adjust(Base __y, 3, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
```

```
modify(draw __rectangle, Base __x, Base __y,
                        Args, Arguments, Object) :- !,
    stack __vector(Args, Length),
    new __stack __vector(Arguments, Length),
    adjust(Base __x, 0, Args, Arguments),
    adjust(Base __y, 1, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
modify(draw __circle, Base__x, Base __y,
                        Args, Arguments, Object) :- !,
    stack __vector(Args, Length),
    new __stack __vector(Arguments, Length),
    adjust(Base __x, 0, Args, Arguments),
    adjust(Base __y, 1, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
modify(draw __arc, Base __x, Base __y,
                        Args, Arguments, Object) :- !,
    stack __vector(Args, Length),
    new __stack __vector(Arguments, Length),
    adjust(Base __x, 0, Args, Arguments),
    adjust(Base __y, 1, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
modify(draw __fan, Base __x, Base __y,
                        Args, Arguments, Object) :- !,
    stack __vector(Args, Length),
    new __stack __vector(Arguments, Length),
    adjust(Base __x, 0, Args, Arguments),
    adjust(Base __y, 1, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
```

```
modify(draw__ellipse, Base__x, Base__y,
                        Args, Arguments, Object) :- !,
    stack__vector(Args, Length),
    new__stack__vector(Arguments, Length),
    adjust(Base__x, 0, Args, Arguments),
    adjust(Base__y, 1, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
modify(draw__elliptical__arc, Base__x, Base__y,
                        Args, Arguments, Object) :- !,
    stack__vector(Args, Length),
    new__stack__vector(Arguments, Length),
    adjust(Base__x, 0, Args, Arguments),
    adjust(Base__y, 1, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
modify(draw__graph, Base__x, Base__y,
                        Args, Arguments, Object) :- !,
    adjusts(Base__x, 0, 1, Args, Arguments0),
    adjusts(Base__y, 0, 2, Arguments0, Arguments)
    ;
```

```
modify(draw __polygon, Base __x, Base __y,
                        Args, Arguments, Object) :- !,
    adjusts(Base__x, 0, 1, Args, Arguments0),
    adjusts(Base__y, 0, 2, Arguments0, Arguments)
    ;
modify(draw __filled__rectangle, Base __x, Base __y,
                        Args, Arguments, Object) :- !,
    stack __vector(Args, Length),
    new __stack __vector(Arguments, Length),
    adjust(Base __x, 0, Args, Arguments),
    adjust(Base __y, 1, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
modify(draw __filled__circle, Base__x, Base __y,
                        Args, Arguments, Object) :- !,
    stack __vector(Args, Length),
    new __stack __vector(Arguments, Length),
    adjust(Base __x, 0, Args, Arguments),
    adjust(Base __y, 1, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
modify(draw __filled__ellipse, Base __x, Base __y,
                        Args, Arguments, Object) :- !,
    stack __vector(Args, Length),
    new __stack __vector(Arguments, Length),
    adjust(Base __x, 0, Args, Arguments),
    adjust(Base __y, 1, Args, Arguments),
    copy__element(Length-1, Args, Arguments)
    ;
```

```
modify(draw__filled__fan, Base__x, Base__y,
                        Args, Arguments, Object) :- !,
    adjusts(Base__x, 0, 1, Args, Arguments0),
    adjusts(Base__y, 0, 2, Arguments0, Arguments)
    ;
modify(draw__characters, Base__x, Base__y,
                        Args, Arguments, Object) :- !,
    adjusts(Base__x, 2, 1, Args, Arguments0),
    adjusts(Base__y, 2, 2, Arguments0, Arguments1),
    stack__vector(Args, Length),
    new__stack__vector(Arguments, Length),
    vector__element(Args, 2, Font__name),
    get__font(Object!font__bank, Font__name, Font, Object),
    vector__element(Arguments, 2, Font),
    copy__element(Length-1, Arguments1, Arguments)
    ;
```

```
%  get __font/4
%        フォントのパス名からフォント・オブジェクトを生成する。生成した
%      フォント・オブジェクトはフォント・バンク（クラスlist__index のイ
%      ンスタンス）に保持しておく。
get __font(0, Font__name, Font, Object):- !,
    :create(#list __index, Font __bank),
    Object!font __bank := Font__bank,
    :create(#font, Font __name, Font),
    :add__at(Font __bank, Font, Font__name)
    ;
get __font(Font __bank, Font__name, Font, Object) :-
    :get__at(Font __bank, Font, Font__name), !,
    ;
get __font(Font __bank, Font__name, Font, Object) :-
    :create(#font, Font __name, Font),
    :add__at(Font __bank, Font, Font__name)
    ;

%  adjust/4
%        ソース引数ベクタArgsのPosition番目の要素にベース値Baseを足して、
%      引数ベクタArguments のPosition番目の要素とユニファイする。
%        ベース値が０の場合は何もしない。
adjust(0, Position, Args, Arguments) :- ! ;
adjust(Base, Position, Args, Arguments) :-
    vector__element(Args, Position, Current),
    New = Current + Base,
    vector__element(Arguments, Position, New)
    ;
```

```
%  adjusts/5, adjusts/4
%        ソース引数ベクタArgsのPosition番目の要素であるPrologリストの各
%      要素（スタック・ベクタ）Infoの V__position番目の要素にベース値Ba
%      seを足し、引数ベクタArguments の V__position番目の要素とユニファ
%      イし、他要素を copy __element/3 により Argumentsにコピーする。
adjusts(0, Info __position, V __position, Args, Arguments) :- ! ,
    Arguments = Args
    ;
adjusts(Base, Info__position, V __position, Args, Arguments) :-
    stack __vector(Args, Length),
    vector__element(Args, Info__position, Current __info),
    adjusts(Current __info, V __position, Base, New __info),
    new __stack __vector(Arguments, Length),
    vector__element(Arguments, Info __position, New __info),
    copy__element(Length-1, Args, Arguments)
    ;

adjusts([], Position, Base, []) :- ! ;
adjusts([Info | Rem], Position, Base, [New__info | New]) :-
    adjusts(Rem, Position, Base, New),
    stack __vector(Info, Length),
    new __stack __vector(New__info, Length),
    vector__element(Info, Position, Current),
    New __value = Current + Base,
    vector__element(New __info, Position, New __value),
    copy__element(Length-1, Info, New __info)
    ;
```

```
%  copy__element/3
%        スタック・ベクタ（Source）の要素を別のスタック・ベクタ（Dest）
%      にコピーする。
%        コピーはユニフィケーションによって行い、未定義変数の要素にだけ
%      コピーする。
copy__element(Position, Source, Dest) :-
    Position < 0, !
    ;
copy__element(Position, Source, Dest) :-
    vector__element(Source, Position, Element),
    vector__element(Dest, Position, Element), !,
    copy__element(Position-1, Source, Dest)
    ;
copy__element(Position, Source, Dest) :-
    copy__element(Position-1, Source, Dest)
    ;
```

```
%  data/2
%        描画メソッドのメソッド名と引数の情報を有する。
%        第１引数がメソッド名、第２引数がメソッドの引数情報を要素とする
%      スタック・ベクタ。
%        フォントはパス名を表す２バイト・ストリングで記述されている。
data(draw __line, {20, 30, 250, 300, 2, solid, set}) ;
data(draw __rectangle, {100, 50, 300, 200, 2, solid, set}) ;
data(draw __circle, {200, 200, 100, 1, dotted, 2}) ;
data(draw __arc, {100, 100, 150, 280, 40, 3, solid, 3}) ;
data(draw __fan, {300, 300, 150, 95, 170, 3, solid, 4}) ;
data(draw __ellipse, {300, 300, 150, 100, 1, solid, 5}) ;
data(draw __elliptical__arc, {100, 100, 50, 70, 200, 30, 1, solid, 2}) ;
data(draw __graph, {[(10, 10), (200, 50), (50, 300)], 0, 0, 400, 400, 4, solid, 3})
;
data(draw __polygon, {[(10, 10), (200, 50), (50, 300)], 0, 0, 400, 400, 4, solid, 4}
) ;
data(draw __filled__rectangle, {100, 50, 300, 200, 1, mesh, 5}) ;
data(draw __filled__circle, {200, 200, 100, 1, left __hatch, set}) ;
data(draw __filled__ellipse, {200, 0, 100, 1, dotted, set}) ;
data(draw __filled__fan, {300, 300, 150, 95, 170, 1, small__mesh, 3}) ;
data(draw __characters, {[{16#'2341', 50, 50, 50, 100}], 2, 'font __13'}) ;
data(draw __characters, {[{16#'3441', 100, 50, 50, 100}, {16#'3B7A', 150, 50, 50
, 100}], 3, '>sys>font>kanji __16.font'}) ;

end.
```