〔非公開〕

ＴＲ－Ｃ－００３９

図形画記述／検索　ＳＰＡＤＥ　ＳＹＳＴＥＭ

インターフェース・ツール

島　則之
NORIYUKI SHIMA

高橋　友一
TOMOICHI TAKAHASHI

１９９０．１．１０

ＡＴＲ通信システム研究所

図形画記述 / 検索 SPADE SYSTEM
インターフェース・ツール

島 則之 高橋友一
(株) ＡＴＲ通信システム研究所
知能処理研究室

1月8日, 1990

目次

1 SPADE SYSTEM インターフェース・ツール群

1.1 概要

本章では先ず、図形画記述／検索システムである SPADE SYSTEM (SPAtial-relationship based DEscription/retrieval SYSTEM) に対するインターフェース・ツール群について簡単な説明を行なう. 図 1 に簡単にインターフェース・ツール群と SPADE SYSTEM との関連を示す. SPADE SYSTEM の詳細については, 東洋情報システム (株) 納入の説明書を参照して下さい.

1.2 機能

画像内容検索ツール hokusai

SUN の suntool 上でグラフィック・インターフェース・ウィンドウを表示し, 円もしくは多角形で画像を描き, その画像に似た画像を検索するよう SPADE SYSTEM に起動をかける.

輪郭線抽出ツール pictrace

画像ファイル (ビットマップイメージ<rasterfile>) を表示し, マウス操作により画像内の任意の物体の輪郭線を円もしくは多角形で切り出し、SPADE 用の入力データを作成する.

データ変換ツール sha2art, art2sha

お絵書きツール『写楽』[1]により作成された画像ソースプログラムを SPADE 用データへ変換する. 及びその逆変換を行なう.

画像表示ツール image-server

SPADE SYSTEM からの起動で, お絵書きツール『写楽』の絵もしくはビットマップイメージ<rasterfile>を SUN の suntool 上に表示する.

1.3 ハードウェア環境

SUN3/260

1.4 ソフトウェア環境

- SUN/JNIX4.2BSD/3.4EXPORT
- SUN Common-Lisp(Ver 2.1.3)
- SUN Parser(Ver 2.0)
- SHARAKU DRAWING TOOL

[1] 『写楽』は ASTEC 社の登録商標です.

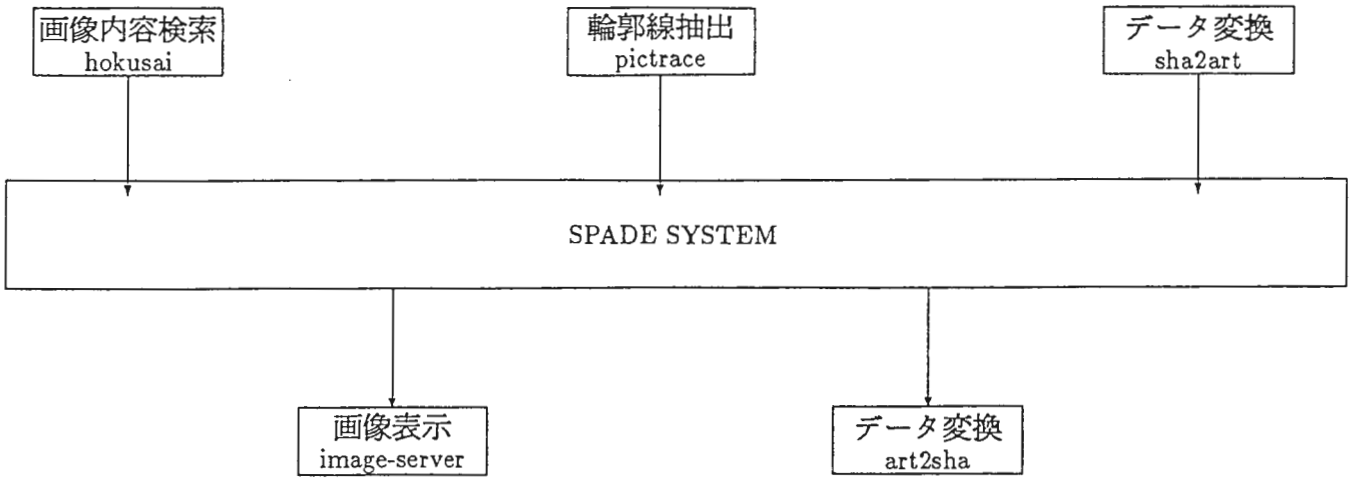


図1 SPADE SYSTEM と インターフェース・ツール群

2 写楽 ←→ SPADE フィルター

2.1 処理概要

お絵書き用ツール『写楽』（以後『写楽』）で作成した図形画のCソースプログラムをSPADE用データファイルへ変換する．もしくはその逆変換を行なう．
SPADEでの図形画の表現は付録Cを参照して下さい(詳しくは東洋情報システム(株)納品の図形画位置記述プログラム・システム説明書を参照して下さい)．

1. 『写楽』 → SPADE （コマンド名 sha2art）

   (a) sha2artでは『写楽』で作成した図形画のCソースプログラムからSPADE SYSTEM入力用フォーマット(付録C)のサブセットである以下の

   ```
   <figure-data> ::=
       (dat position <obj> <position> <cf>) |
       (dat shape <obj> <shape> <cf>) |
       (dat tilt <obj> <tilt> <cf>) |
       (dat attribute <obj> <attribute> <cf>) |
   ```

   のフォーマットに変換する．

   (b) <shape>の値はcircleもしくはpolygonとする．

   (c) <tilt>の値は全て0.0とする．

   (d) <obj>の値として，順次A1, A2, A3, . . . のように与える．

   (e) attributeに関しては、『写楽』で図形画の作成時にattributeを指定できない為に<attribute>値は<obj>値と同値になっている．
   （例） (dat attribute A1 A1 1)

2. SPADE → 『写楽』 （コマンド名 art2sha）
   SPADEでは現在色データを使用していない為，SPADEデータから『写楽』用Cソースプログラムへ変換するとエッジだけに色のある画像となる．

2.2 実行方法

1. sha2art セグメント番号
   （例） sha2art 5

   (a) セグメント番号とは『写楽』で画像を作成する時に指定するための画像の番号である．

   (b) この例の実行によりseg5.cファイルを入力ファイルとし，fig5.artファイルを出力ファイルとして上記の変換を行なう．

   (c) 従ってseg##.c及びfig##.artというファイル名称は本フィルターでは固定としている．（##はセグメント番号である）

   (d) seg##.cファイルは/usr0/sharaku/productの下のファイルを仮定している．

   (e) fig##.artファイルは/usr0/sharaku/productの下に出力される．

2. art2sha　入力ファイル　出力ファイル
　　(例)　art2sha in.dat out.c

(a) in.dat SPADE 用データファイルが out.c なる C ソースプログラムに変換される.

(b) out.c をコンパイル、実行することにより、SPADE で扱っていた画像を見ることが出来る (コンパイル方法は 2.5 を参照して下さい).

(c) 入力ファイル、出力ファイルの名称は任意である.

2.3　ソースファイル

1. sha2art
　　CS08: /usr3/shigeru/sharaku-to-art/src/sharaku-to-art.1.05.c

2. art2sha
　　CS08: /usr3/shigeru/art-to-sharaku/src/art2sha.1.00.c

2.4　言語

1. sha2art …… C 言語

2. art2sha …… C 言語

2.5　写楽 追記

1. 『写楽』は ASTEC 社のお絵書きツールである. 使用法については「写楽使用の手引」を参照して下さい.

2. cykernel.wiss に『写楽』で作成した画像ファイルがすべて格納される.

3. cykernel.wiss は『写楽』を起動したディレクトリに作成される.

4. cykernel.wiss は画像をセグメント番号で管理している.

5. 『写楽』で作成した cykernel.wiss に dewiss というコマンドを実行することにより C ソースプログラムが作成される.
　　dewiss 実行方法
　　dewiss -W cykernel.wiss -m SUNWINDC -l ##

(a) ## はセグメント番号である.

(b) これを実行することにより seg##.c という C ソースプログラムが作成される.

6. 作成された seg##.c ファイルをコンパイルすることにより, SUN ウィンドウ上にその画像を表示することができる.

7. 但し, 作成された C ソースプログラムを直接コンパイルするとエラーとなるため一部修正を行なう必要がある.

修正箇所

1. `/* include standard header files */` 行の次行

```
/* include standard header files */      /* include standard header files */
include <stdio.h>                         include <stdio.h>
include <gksuser.h>                  →    include <gksuser.h>
                                          include <gksname.h>
                                          include "set_col_rep"
```

2. /* start of seg?? */ 行の次行

```
/* Set default attributes */                    /* Set default attributes */
SET_ASPECT_SOURCE_FLAGS(initasf);      →        set_col();
                                                SET_ASPECT_SOURCE_FLAGS(initasf);
```

3. /* Main program */ 行の次行

```
OPEN_GKS("/dev/tty/", 1000);                    OPEN_GKS("GKS.error", 10000);
OPEN_WORKSTATION(1, "/dev/tty",          →      OPEN_WORKSTATION(1, "",
                        "SUNWINDC");                            "SUNWINDC");
```

8. コンパイル方法は以下の通り.

コンパイル方法

```
cc -f68881 -o seg## seg##.c -O -lcykernel -lcgi -lsuntool -lsunwindow -lpixrect
                                                                        -lm
```

(a) コンパイルを行なう時, ソースファイルの修正で追加をした include "set_col_rep"の set_col_rep ファイルはコンパイルするディレクトリになければならない.

(b) set_col_rep ファイルは CS08 の /usr0/sharaku/product に存在する.

(c) これで seg## というコマンドで実行ができる.

3 HOKUSAI

3.1 処理概要

SUN のグラフィック・ウィンドウ上で円もしくは多角形を描き，その画像に似た画像を検索するよう SPADE SYSTEM に起動をかける.

3.2 処理構成

hokusai は次の 3 つのプロセスより構成されている.

1. hokusai
2. `lisp_server`
3. feature

3.3 各プロセス説明

図 2 に各プロセスと SPADE SYSTEM との関係を示す.

1. hokusai
   (a) グラフィック・ウィンドウを提供し，ウィンドウ上で作成された画像を SPADE 用画像ファイルとして出力を行う.
   (b) 画像データファイルが出力されたことを `lisp_server` に知らせる（信号を送る）クライアント部分である.
2. `lisp_server`
   (a) プロセス間通信のサーバー部分である.
   (b) hokusai より信号を受け取り、受け取った信号を SPADE SYSTEM 側へ送信する.
3. SPADE SYSTEM 側 (feature)
   (a) 常に `lisp_server` からの信号を待ち、信号を受け取ると、画像データファイルを読み込み、位置関係を求め、そして評価値 (cf) の最も高い関係を抽出する.
   (b) 抽出された関係と同じ関係を持つ画像を SPADE の nl-search コマンドにより検索する.

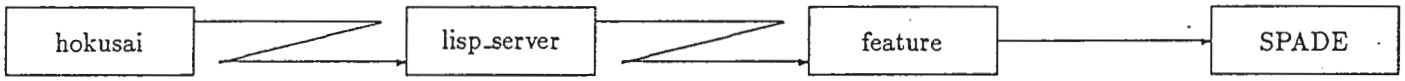


図 2 SPADE SYSTEM と各プロセス

3.4 実行方法

1. `hokusai & <CR>` ＊任意のウィンドウ上で行なう＊
2. `lisp_server & <CR>` ＊任意のウィンドウ上で行なう＊
3. `(feature) <CR>` *SPADE オペレーションウィンドウ上で行なう＊
   もしくは SPADE SYSTEM 側で、
   `(sample-sharaku)`
   として，メニューを表示させ, 10 番の Hokusai を選択する.

3.5 操作方法

1. POLYGON ボタン

   - 多角形を描くときに指定する.
   - 最初にマウスの左ボタンを押した箇所を始点とし、それ以降は左ボタンを押した点と一つ前に押された点とを直線で結ぶ.
   - そして、中央ボタンを押すと最後に左ボタンの押された点と、始点とを直線で結ぶ.
   - 中央ボタンを押すまでに必ず属性（ATTRIBUTE） の指定をしておかなければならない.

   ※注　多角形は必ず左周りに描かなければならない.

2. CIRCLE ボタン

   - 最初に押された点を円の中心とし、次に押された点を円周上の一点として円を描く.
   - 円周上の一点を指定するまでに必ず ATTRIBUTE を指定しなければならない.

3. att フィールド

   - 作成する図形の attribute を決める (付録 C 参照の事).

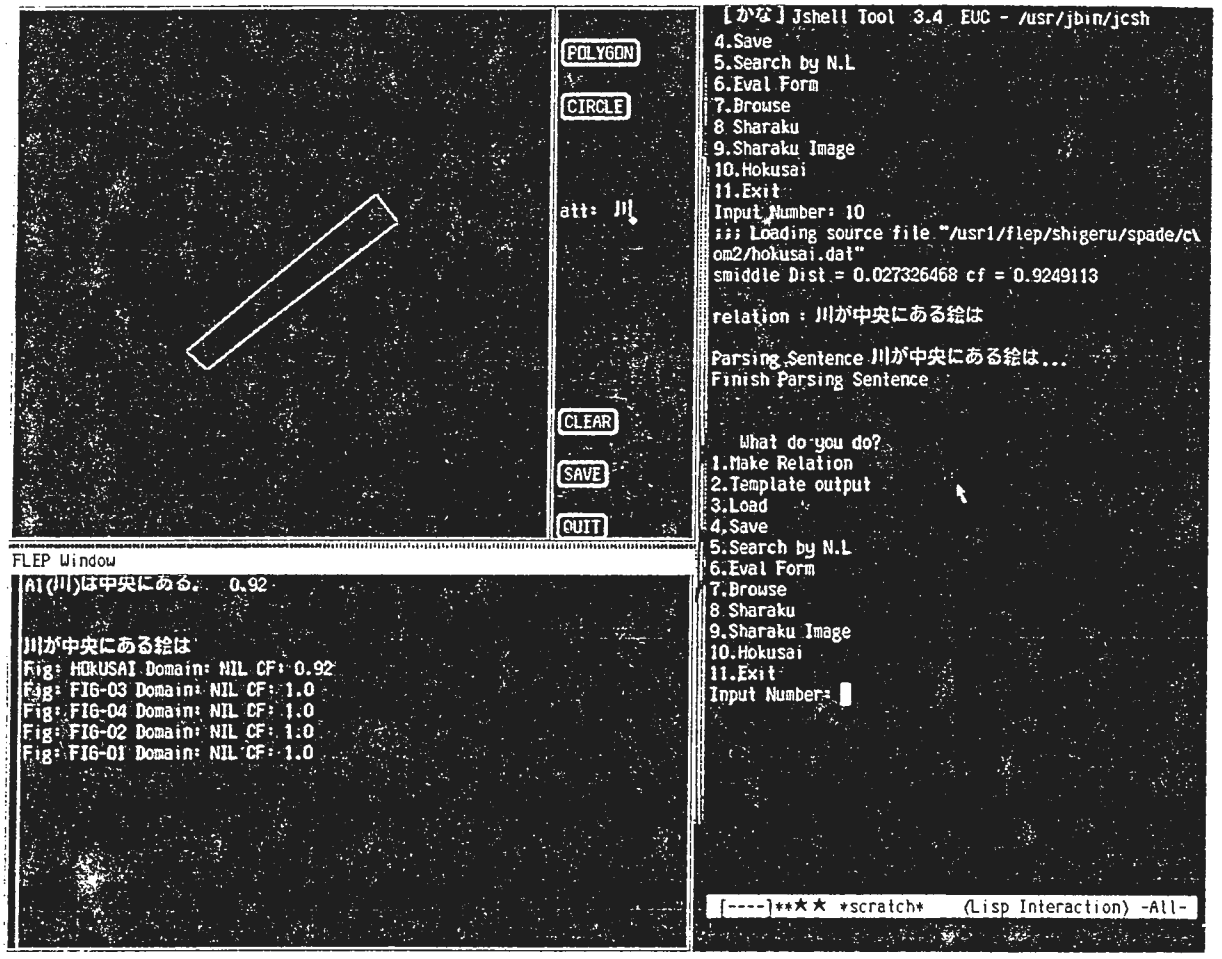


図 3 HOKUSAI 画面 (左上)

4. CLEAR ボタン

   - 保持している OBJECT 情報及び画像を消去する.

5. SAVE ボタン

   - 作成された円、多角形の OBJECT 情報をファイル (hokusai.dat) へ出力する.

6. QUIT ボタン

   - 処理を終了する.

3.6 ソースファイル

1. hokusai
   `CS08 : /usr1/flep/shigeru/spade/com2/hokusai.c`

2. `lisp_server`
   `CS08 : /usr1/flep/shigeru/spade/com2/lisp_server.c`

3. feature
   `CS08 : /usr1/flep/shigeru/spade/com2/feature.lisp`
   `CS08 : /usr1/flep/shigeru/spade/com2/l_client.c`

4. Makefile
   `CS08 : /usr1/flep/shigeru/spade/com2/Makefile`

3.7 言語

1. hokusai.c ····· C 言語

2. `lisp_server.c` ····· C 言語

3. s-feature.lisp ····· Lisp

4. `l_client.c` ····· C 言語

4 PICTRACE

4.1 処理概要

新しいウィンドウを作り、そのウィンドウ上に指定した画像（ビットマップイメージ）を表示する．その画像上でマウスを操作することにより、画像内の任意の物体の輪郭線の抽出を行なうことができる．そしてＳＰＡＤＥ用データファイルが作成される．

4.2 実行方法

pictrace　画像ファイル名
（例）　pictrace shima1.tmp3

4.3 操作方法

1. POLYGON ボタン

- 多角形を描くときに指定する．
- 最初にマウスの左ボタンを押した箇所を始点とし、以降は左ボタンを押すと一つ前に押された点とを直線で結ぶ．
- 中央ボタンを押すと、一つ前に押された点と始点とを直線で結ぶ．
- 中央ボタンを押すまでに必ず属性（ATTRIBUTE） の指定をしておかなければならない．

（注）　多角形は必ず左周りに描かなければならない．

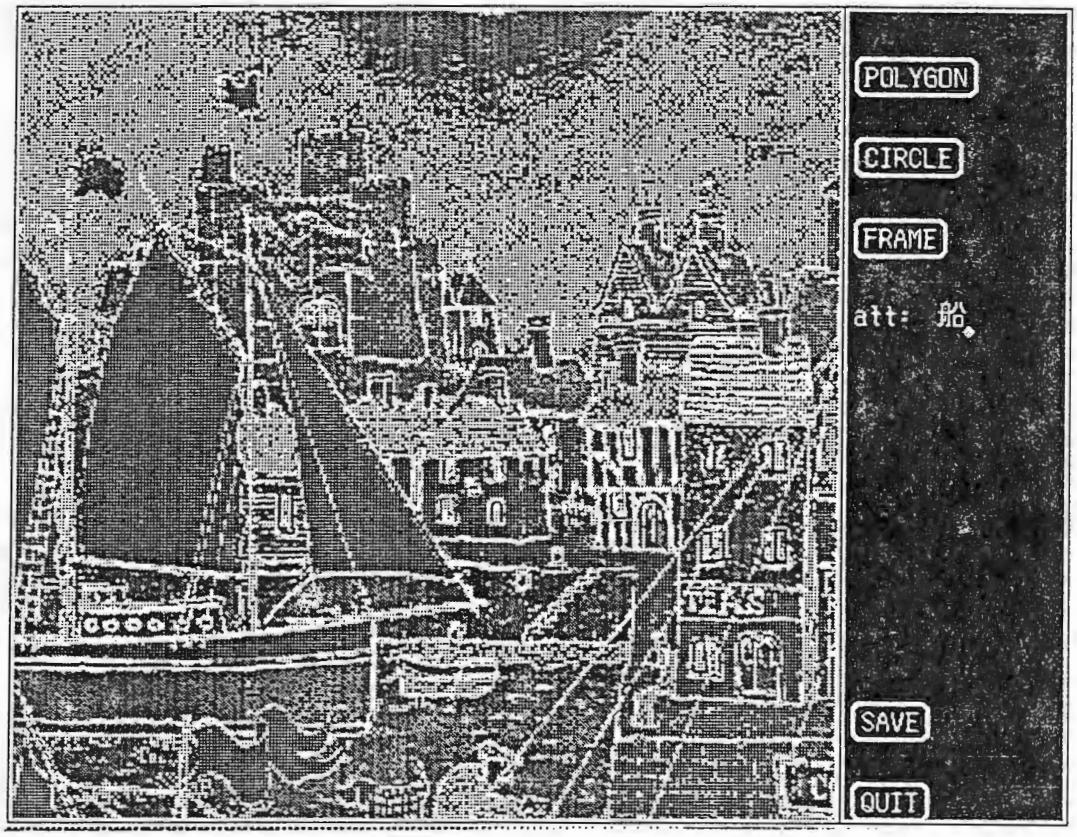


図 4 PICTRACE 画面

2. CIRCLE ボタン

   - 最初に押された点を円の中心とし次に押された点を円周上の一点として円を描く.
   - 円周上の一点を指定するまでに ATTRIBUTE を指定しなければならない.

3. FRAME ボタン

   - 画像の枠を決める.
   - 枠となる長方形の対角の 2 点をマウスの左ボタンまたは中央ボタンで指定する.
   - 指定された 2 点を対角の頂点とする長方形を画像上に描き、その長方形の左下の座標が (0, 0) となり、右上の座標が (1.0, 1.0) となる.

4. att フィールド

   - 作成する図形の attribute を決める (付録 C 参照の事).

5. SAVE ボタン

   - 作られた円・多角形の OBJECT 情報をファイルへ出力する.
   - 出力されるファイル名は画像ファイル名に拡張子 (.) が付いていれば拡張子の後に .dat を、拡張子がなければ画像ファイル名の後に .dat をつける.

6. QUIT ボタン

   - 処理を終了する.

4.4 ソースファイル

1. pictrace

   `CS08 : /usr3/shigeru/pictrace/pictrace.c`

2. Makefile

   `CS08 : /usr3/shigeru/pictrace/Makefile`

4.5 言語

C 言語

5 IMAGE-SERVER

5.1 処理概要

SPADE の画像検索機能により検索された画像の中から、見たい画像を指定することにより、他のウィンドウにその画像を表示する.

5.2 処理構成

IMAGE-SERVER は次の 2 つのプロセスより構成されている.

1. image-server
2. sharaku-image

5.3 各プロセス説明

図 5 に各プロセスと SPADE SYSTEM との関係を示す.

1. image-server
   (a) グラフィック・ウィンドウを提供し, SPADE 側で指定した画像をウィンドウ上に表示する.
   (b) プロセス間通信のサーバー部分である.
2. SPADE SYSTEM 側 (sharaku-image)
   (a) SPADE の画像検索機能により得られた画像を/usr1/flep/usr/window.datファイルから読み込む.
   (b) 読み込んだ画像のリストを表示し, 選択用のプロンプトを表示する.
   (c) 指定された画像を image-server に伝える.
   (d) プロセス間通信のクライアント部分である.

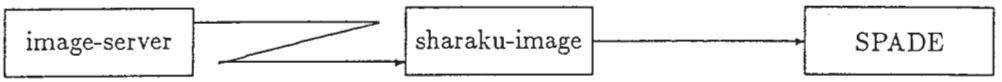


図 5 SPADE SYSTEM と各プロセス

5.4 実行方法

1. 画像を表示したいウィンドウで
   ```
   image-server <CR>
   ```
2. SPADE オペレーション・ウィンドウ上で
   ```
   (sharaku-image)<CR>
   ```
   とするか, もしくは SPADE SYSTEM 側で、
   ```
   (sample-sharaku)<CR>
   ```
   として, メニューを表示させ, 9 番の Sharaku-image を選択する.

※注

- image-server 側で表示する画像ファイルが『写楽』ファイルの場合、/usr1/flep/imgの下になければならない. ファイル名はfig##でないといけない.
- image-server 側で表示する画像ファイルが浮き世絵ファイルの場合、/usr1/flep/img2の下になければならない. ファイル名はshima##でないといけない.
- 選択できる画像ファイルは、/usr1/flep/usr/window.dat ファイルより抽出されるため Sharaku-image を実行する直前に nl-search を行ない、window.dat ファイルへ nl-search の結果を出力しておかなければならない.(window.dat ファイルの内容は、nl-search の結果でなければならない.)

sharaku-image オペレーション 例

```
>(sample-sharaku)

What do you do?
1.Make Relation
2.Template output
3.Load
4.Save
5.Search by N.L
6.Eval Form
7.Browse
8.Sharaku
9.Sharaku Image
10.Hokusai
11.Exit
Input Number: 9                    ** 9番を選択する **

"山のふもとを川が流れる絵は "     **nl-search で検索した画像の特徴を表示する **

1.FIG-02                           ** 検索された画像ファイル名 **
2.FIG-01
input seg-file number : 2          ** 見たい画像の番号を選択する **
```

window.dat ファイル

```
% cat window.dat

山のふもとを川が流れる絵は

Fig: FIG-02 Domain: NIL CF: 0.97
Fig: FIG-01 Domain: NIL CF: 0.97

%
```

5.5 ソースファイル

1. image-server
   CS08 : /usr1/flep/shigeru/spade/com/image-server.c
2. sharaku-image
   CS08 : /usr1/flep/shigeru/spade/com/client2.c
   CS08 : /usr1/flep/shigeru/spade/sample.lisp

3. Makefile

   `CS08 : /usr1/flep/shigeru/spade/com/Makefile`

5.6 言語

1. image-server.c ····· C 言語
2. client2.c ····· C 言語
3. sample.lisp ····· Lisp

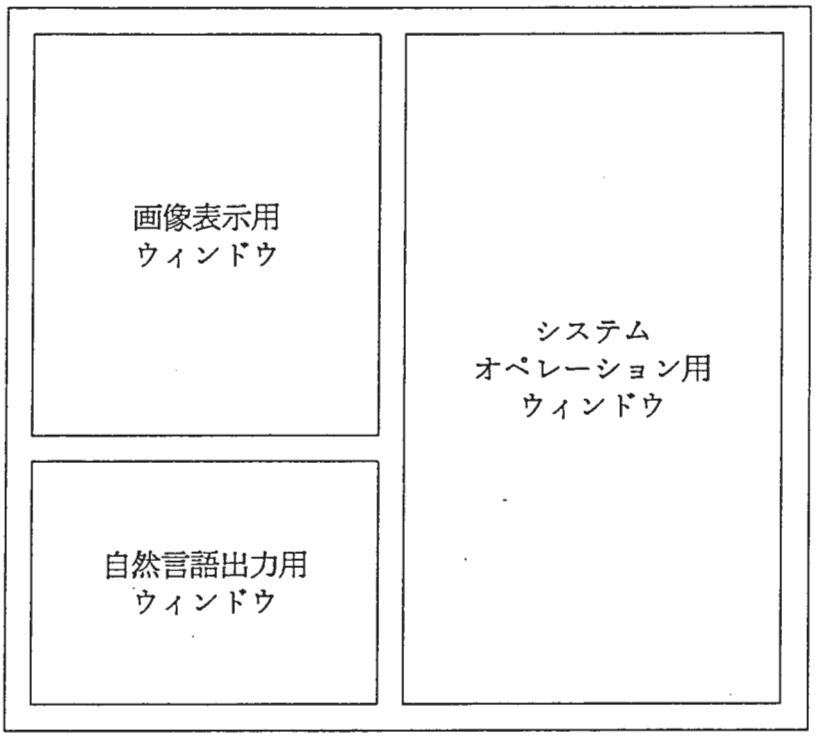
A　付録 SPADE SYSTEM での環境設定

本付録 A で，今までのインターフェース・ツールを SPADE SYSTEM と一緒にどのようにして使用するかについて主に説明を行なう．

A.1　セットアップ方法

1. 画像表示用ウィンドウとシステムオペレーション用ウィンドウを作る.(Jshell Tool 使用の事)
2. システムオペレーション用ウィンドウで次の操作を行なう．

```
% com_man& <CR>      ** サーバ起動 **
% flep_wdw& <CR>     ** 自然言語出力用ウィンドウが作成される **
% cd /usr1/flep/shigeru/spade <CR>
                     ** ディレクトリの移動 **
% lisp <CR>          ** ＬＩＳＰ起動 **
> (load "setup") <CR>
                     ** セットアッププログラムをロード **
  Memory Expand OK? (Y or N) y <CR>
  Load Parser OK? y <CR>
  Load Comunication System (Y or N) y <CR>
                     ** ロード途中の問い合わせはすべて y で答える **
> (in-package 'flep) <CR>
```



SPADE SYSTEM 画面レイアウト

3. 画像表示用ウィンドウで次の操作を行なう.

```
% cd /usr1/flep/shigeru/spade/com <CR>
                    ** ディレクトリの移動 **
% image-server & <CR>
                    ** 画像表示用サーバプログラム起動 **
```

A.2 サンプルプログラム操作法

操作はシステムオペレーション用ウィンドウで行なう. セットアップが完了した状態で (sample-sharaku) を起動する.

```
> (sample-sharaku) <CR>
```

システムオペレーション用ウィンドウ上に

```
   What do you do?

1.Make Relation
2.Template output
3.Load
4.Save
5.Search by N.L
6.Eval Form
7.Browse
8.Sharaku
9.Sharaku Image
10.Hokusai
11.Exit
Input Number:
```

とメニュー画面が表示される.

1. Make Relation

関係付けを行いたいレベルの選択を行う.

```
Select Level
1.Level 3
2.Level 4
3.Level 3 & 4
4.Cancel
Input Number: 1          ** レベル3を選択 **
```

データベースにロードされている SPADE 用データの一覧が表示されるので,
関係付けを行いたいデータを選択する.

```
Select Fig-id
1.SHIMA1
2.SHIMA2
3.SHIMA3
4.SHIMA4
```

```
5.SHIMA5
6.SHIMA6
Input Number: 1          ** SHIMA1 を選択 **
```

選択したデータの関係付けが行なわれる.

```
Evaluate Form : (DEF-MAKE-REQUIRE (REL (?LEVEL (= ?LEVEL 3)) ? ? ? ? SHIMA1 ?))
                          ** 関係付け実行中のメッセージ **
```

関係付けが完了するとメニュー画面に戻る.

2. Template output

自然言語出力を行ないたいレベルを選択する.

```
Template output
1.Level 3
2.Level 4
3.Level 3 & 4
4.Cancel
Input Number: 1          ** レベル 3 を選択 **
```

データベースにロードされている SPADE 用データの一覧が表示されるので,
自然言語出力を行ないたいデータを選択する. この時選択するデータはすでに
Make Relation で関係付けが行なわれていなければならない.

```
Select Fig-id
1.SHIMA1
2.SHIMA2
3.SHIMA3
4.SHIMA4
5.SHIMA5
6.SHIMA6
Input Number: 1          ** SHIMA1 を選択 **
```

選択したデータの自然言語への変換、出力が行なわれる.

```
Evaluate Form : (DEF-PRINT-REQUIRE (REL (?LEVEL (= ?LEVEL 3)) ? ? ? ? SHIMA1 ?\))
                          ** 自然言語実行中のメッセージ **
```

自然言語出力が完了するとメニュー画面に戻る.

3. Load

データベースにロードしたい SPADE 用データファイル名を入力する.

```
Input File Name: hokusai.dat    ** hokusai.dat をロードする **
Input File Name: <CR>           ** RETURN で入力終了 **
```

SPADE用データファイルがロードされる.

```
;;; Loading source file "hokusai.dat"
```

ロードが完了するとメニュー画面に戻る.

4. Save
関係を求めた図形の情報をセーブする. 詳しくはSPADE システム説明書を参照して下さい.

5. Search by N.L

自然言語問い合わせ方法を選択する.

```
1.Input
2.Selection
3.Cancel
Input Number:
```

(a) 1を選択した時
問い合わせしたい事柄を自然言語で入力する.

```
Input Sentence: 湖が山と家の間にある絵は    ** 自然言語で入力 **
```

(b) 2を選択した時
問い合わせしたい事柄を選択する.

```
1. 湖が山と家の間にある絵は
2. 山のふもとを川が流れる絵は
3. 月が湖を照している絵は
4. 上に太陽があり山のふもとを川が流れる絵は
5. 湖が山と川と家の間にあり山のふもとを川が流れる絵は
6. 川が山のふもとを流れ月が山のかなり上方にある絵は
7. 山と家が向い合った絵は
8. 山と山が向い合った絵は
9. 湖が山の間にある絵は
10. 山のふもとに家がある絵は

Input Number: 1       ** "湖が山と家の間にある絵は"を選択  **
```

自然言語の問い合わせに対する画像の検索が行なわれ、自然言語出力用ウィンドウに出力される.

```
Evaluate Form ... (nl-search "湖が山と家の間にある絵は")
Parsing Sentence 湖が山と家の間にある絵は...
Finish Parsing Sentence
```

出力が完了するとメニュー画面に戻る.

6. **Eval Form**
   LISP 環境でコマンド入力を行なう.

7. **Browse**

   見たいデータベースの内容を選択する.

```
1.Brwose fact
2.Browse rule
3.Browse strategy
4.Cancel
Input Number:
```

(a) 1 を選択した時
   見たいレベルの関係を選択する.

```
1.Dat all
2.Rel all
3.Rel level 0
4.Rel level 1
5.Rel level 2
6.Rel level 3
7.Rel level 4
8.Cancel
Input Number: 3        ** レベル 0 を選択 **
```

データベースにロードされているデータの一覧が表示されるので、
見たいデータを選択する.

```
Select Fig-id
1.SHIMA6
2.SHIMA5
3.SHIMA4
4.SHIMA3
5.SHIMA2
6.SHIMA1
Input Number: 1        ** SHIMA6 を選択 **
```

データベースに記録されているデータが表示される.

(b) 2 を選択した時
   見たいルールの選択方法を選択する

```
1.Select by definition
2.Select by level & relname
```

```
3.All
4.Cancel
Input Number:
```

(ア) 1を選択した時

```
1.Enter definition-id
2.Select definition-id
3.Cancel
Input Number:
```

(i) 1を選択した時
ルール名を入力する.

```
Enter definition-id: 照す-definition     ** ルール名入力 **
```

(ii) 2を選択した時
見たいルールを選択する.

```
Select definition-id
1. 照す -DEFINITION
2. 川沿い -DEFINITION
3. 湖岸 -DEFINITION
4. 手前 -DEFINITION
5. ふもと -DEFINITION
6. もより -DEFINITION
        :
        :
Input Number: 1              ** 照す -DEFINITION を選択 **
```

(ィ) 2を選択した時
不明

(ゥ) 3を選択した時
すべてのルールを表示する.

(c) 3を選択した時
不明

表示が完了するとメニュー画面に戻る.

8. Sharaku
sharaku プログラムとのプロセス間通信により受け取ったファイル名のファイルをロードし、位置関係の作り込みを行ない、そして関係付けられた結果を自然言語で自然言語出力用ウィンドウに表示する. そしてメニュー画面に戻る.

9. Sharaku Image

Sharaku Image を選択する前に nl-search を実行しなければならない.
見たい画像データを選択する.

”山のふもとを川がながれる絵は”

```
1.FIG-02
2.FIG-01
input seg-file number : 1                    ** FIG-02 を選択 **
```

画像表示用ウィンドウに FIG-02 が表示される.
そしてメニュー画面に戻る.

10. Hokusai

hokusai プログラムとプロセス間通信を行なう.
hokusai 側で save を行なうと描かれた絵の最も特徴的な関係と同じ関係を持つ絵を検索し、自然言語出力用ウィンドウに出力する.

そしてメニュー画面に戻る。

11. Exit
終了する.

B 付録（STOV）

B.1 概要

SUNのラスターファイル（ビットマップイメージ）をＶＩＣＯＭのラスターファイル（ビットマップイメージ）に変換する。

B.2 ラスターファイル 構成

1. SUN

```
COLOR image :  Header                  32 byte
               Color map      red      256 byte
                              green    256 byte
                              blue     256 byte
               Screen image             *** byte

MONO  image :  Header                  32  byte
               Screen image             *** byte
```

2. VICOM

```
COLOR image :  Red   screen image      *** byte
               Green screen image      *** byte
               blue  screen image      *** byte

MONO  image :        Screen image      *** byte
```

```
****************************************
***  MONOCHROME SUN   1 pixel 1 bit  ***
***                                  ***
***             VICOM 1 pixel 8 bit  ***
****************************************
```

- カラー画像の場合SUNはヘッダー32バイト、カラーマップＲ・Ｇ・Ｂそれぞれ256バイトそして画像データ1ピクセル1バイトでnバイトとすると，
  32+3*256+nバイト・サイズの画像ファイルが1つ必要である.
- VICOMは画像データ1ピクセル1バイトでnバイトのサイズのファイルがＲ・Ｇ・Ｂそれぞれ1本ずつの画像ファイルが必要である.

B.3 使用方法

```
% stov 入力ファイル名
```

1. カラーファイルかモノクロファイルかの識別は自動的に行なう.
2. カラーファイルの時は作成されるVICOM用ファイルは、入力ファイル名の後に.r .g .bが付加された3本のファイルとして出力される.
3. モノクロファイルの時は、入力ファイル名の後に .m が付加されたファイルとして出力される.

B.4 ソースファイル

CS08 : /usr3/shigeru/image/stov.1.00.c

B.5 VICOM 操作法

1. CS10 に stov で変換したファイルを転送する.
2. 以下の様に CS10 で操作を行なう.

操作手順

```
% cim                              *** program start ***

> rea 1 ('seg1.r', 2, 1)          *** rasterfile read ***
> rea 2 ('seg1.g', 2, 1)
> rea 3 ('seg1.b', 2, 1)

> two 1                           *** 1 bit shift (1/2) ***
> two 2
> two 3

> red 1>4 (1, 1, 480, 460)        *** 画面サイズ 変更 ***
> red 2>5 (1, 1, 480, 360)
> red 3>6 (1, 1, 480, 360)

> trh 4>7,15 (0, 80)              *** 画面位置移動 ***
> trh 5>8,15 (0, 80)
> trh 6>9,15 (0, 80)

> col 7, 8, 9                     *** R G B 同時出力 ***

> end
```

(注) SUN は輝度が 256 だが VICOM は 128 の為 two を行い輝度を 128 に修正する。

C 付録 図形画表現フォーマット

SPADE SYSTEM での入力図形画は以下のように表現する.

```
<figure-data> ::=
    (dat position <obj> <position> <cf>) |
    (dat shape <obj> <shape> <cf>) |
    (dat color <obj> <color> <cf>) |
    (dat tilt <obj> <tilt> <cf>) |
    (dat attribute <obj> <attribute> <cf>) |
    (rel <level> <name> <obj> <val> <cf>)

<obj>                ::= [symbol]
                           ;;; 図形要素を識別するシンボル (枠は Frame とする)
<position>           ::= (<center-co-ordinate> <one-co-ordinate>) |
                           ;;;shape が circle の場合
                         (<co-ordinate>*)
                           ;;;shape が polygon の場合反時計周りの頂点のリスト
<center-co-ordinate> ::= (<x-center-co-ordinate> <y-center-co-ordinate>)
                           ;;; shape が circle の場合の中心座標
<one-co-ordinate>    ::= (<x-one-co-ordinate> <y-one-co-ordinate>)
                           ;;;shape が circle の場合の円周上の一点座標
<co-ordinate>        ::= (<x-co-ordinate> <y-co-ordinate>)
                           ;;; shape が polygon の場合の頂点座標

<shape>        ::= circle | polygon
<color>        ::= white | black | red | blue ...etc
<tilt>         ;;; 図形の傾き (ラジアン)
<attribute>    ;;; 図形画要素の属性
<cf>           ;;; 確からしさ (-1.0 ～ 1.0)
```