ソフトウェア工学　52 — 1
（１９８７・２・１３）

アイコニック・プログラミング環境 ＨＩ—ＶＩＳＵＡＬ

吉本岩生*，　岩田誠司*，　平川正人**，　田中稔**，　市川忠男**

*広島大学大学院工学研究科　**広島大学工学部

筆者らは既に、データや手続きなどのオブジェクトをアイコンとして画面上に表示し、これによってプログラムを視覚的に作成することができるようなプログラミング言語ＨＩ—ＶＩＳＵＡＬを提案した。本論文では、システム統合機能を持ったアイコンベースのプログラミング環境として提供するためのＨＩ—ＶＩＳＵＡＬの機能拡張について述べる。拡張機能として、（ｉ）プログラム開発やシステム操作のナビゲーション、（ⅱ）オブジェクト指向の概念を取り入れたアイコン・プログラムならびにシステム・オペレーションの解釈／実行メカニズム、（ⅲ）ユーザ定義可能なユーザ・インタフェース、（ⅳ）大規模プログラミングをサポートする入出力データ型制約によるトップダウン・プログラム開発、（ⅴ）既存システムの統合機能、などがある。さらに、実験システムのインプリメンテーションについても述べる。

"An Iconic Programming Environment HI-VISUAL"

Iwao YOSHIMOTO, Seiji IWATA, Masahito HIRAKAWA, Minoru TANAKA, and Tadao ICHIKAWA

Information Systems Lab., Faculty of Engineering, Hiroshima University,
Shitami, Saijo-cho, Higashihiroshima, 724 Japan

We have already proposed a visual programming language named HI-VISUAL with the objective of attaining interactive iconic programming. Programming in HI-VISUAL is carried out simply by arranging icons on the display screen. In this paper, we describe an extension of HI-VISUAL. The extension includes (ⅰ)navigation for the program development and system operations, (ⅱ)interpretation mechanism for icon program and system operation, based on an object-oriented concept, (ⅲ)design of user-defined interfaces, (ⅳ)top-down development of programs, and (ⅴ)integration of existing (sub)systems. Futhermore, we describe design and implementational issues of HI-VISUAL system.

1．　はじめに

コンピュータの普及に伴ってコンピュータの利用者層が拡大し、初心者にも使いやすく、理解しやすいコンピュータ・システムの必要性が高まっている。そのため、対話性の優れたユーザ・インタフェースの研究開発が盛んになってきている。人間とコンピュータとの対話手段として視覚情報を用いるアプローチは最も有望なものの一つである[1]。

ユーザ・インタフェースに視覚情報を導入したシステムとして Xerox Star, Apple Lisa, Macintoshなどがある。それらでは、データやコマンドはアイコンと呼ばれる絵シンボルとして視覚化されており、アイコンを画面上で操作することによって処理を指示することができる。

また、視覚情報を用いてプログラミングを効率的に行なわせることを目的としたシステム[2]~[8]も開発されている。そのようなシステムでは、

ⅰ）アイコンなどによるオブジェクトの視覚化、
ⅱ）フローチャートなどによるアルゴリズムの視覚化、
ⅲ）フォームなどによるデータ構造の視覚化、

が行なわれている。さらに、マルチウインドウやアニメーションを利用したプログラムの多角的表現／動的表現によってユーザのプログラム理解を助けるシステムもある[4][6][7]。

筆者らは、プログラム作成がアイコンを用いて視覚的に行なえるプログラミング言語ＨＩ－ＶＩＳＵＡＬ[9]~[11]を提案した。ＨＩ－ＶＩＳＵＡＬでは、オブジェクトをアイコンで、アルゴリズムをデータフローグラフで、データ構造をアイコンの組合せで視覚化している。ディスプレイ上でアイコンを選択・配置・接続することがプログラミングとなる。

本論文では、システム統合機能をもったアイコンベースのプログラミング環境として提供するためのＨＩ－ＶＩＳＵＡＬの機能拡張について述べる。拡張機能としては（ⅰ）プログラム開発やシステム操作のナビゲーション、（ⅱ）オブジェクト指向の概念を取り入れたアイコン・プログラムならびにシステム・オペレーションの解釈／実行メカニズム、（ⅲ）ユーザ定義可能なユーザ・インタフェース、（ⅳ）大規模プログラミングをサポートするための入出力データ型制約によるトップダウン・プログラム開発、（ⅴ）既存システムの統合機能などがある。

２章では、アイコンの定義ならびにアイコン管理に要求される基本機能について述べる。３章ではアイコニック・プログラミングについて具体的に述べ、それを支援するためのシステムの構成ならびにアイコンの実行メカニズムについて４章で述べる。

2．　アイコニック・システム

2．1　アイコンの定義

アイコンはシステムが扱うファイル、データ、プログラムなどのオブジェクトを表現している。アイコンはインターナル部とエクスターナル部の二つから成っている（表１）。インターナル部はアイコンが示すオブジェクトの意味を記述している。エクスターナル部はオブジェクトの意味の視覚表現を記述しており、それはディスプレイ上に表示される。

ICON　　:= ($I_{internal}$, $I_{external}$)

$I_{internal}$:= (substance, concept, type)
$I_{external}$:= (image, label, shape)

substance　= アイコンの機能記述
concept　　= オブジェクトの概念的名称
type　　= アイコンの型

image　　= オブジェクトを象徴する絵
label　　= アイコンの名前
shape　　= アイコンの枠の形

表１　アイコンの定義

インターナル部はsubstance,concept,typeという３つの属性から成っている。substanceはアイコンが表わしているオブジェクトの実体（データやプログラム）を表している。conceptはオブジェクトの概念的名称（例えば、日付、金額、ソート、コピー）である。これはプログラミング・ナビゲーションやアイコンの分類・検索に利用される。typeはアイコンの型を表わしており、オブジェクトの型によって次の７つがある。（図１）

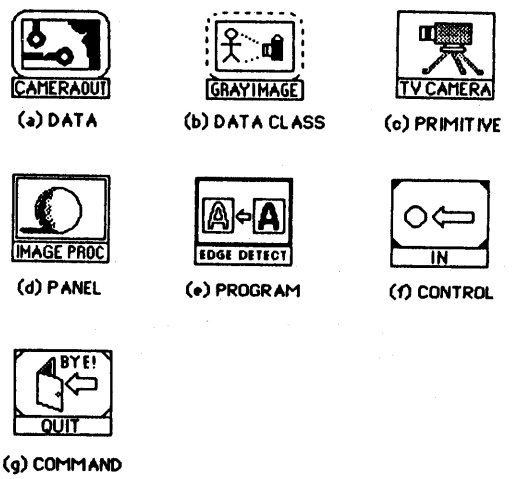

図１　アイコンの型

a）データ・アイコン：文字、数値、図形などの実際のデータを表わすアイコン

b）データクラス・アイコン：データクラスを表わすアイコン

　データクラスはオブジェクト指向の概念におけるクラスに相当し、データクラス間の意味の階層に従って階層化されている。すべてのオペレータ・アイコン（プリミティブ・アイコンとプログラム・アイコン）はデータクラスにメソッドとして記述されている。データクラスの階層構造に従ったメソッドの継承が行なわれるようになっている。（図2）

c）プリミティブ・アイコン：システムにあらかじめ用意される最も基本的なファンクションを表わすアイコン

d）パネル・アイコン：アイコンの集合をひとまとまりにして表わすためのアイコン

　通常のファイルシステムにおけるディレクトリの概念に相当する。

e）プログラム・アイコン：ユーザの作成したプログラム(アイコン・プログラムと呼ぶ)を表わすアイコン

f）コントロール・アイコン：プログラムの実行時に、データの流れを制御する機能（if, loopなど）を表わすアイコン

g）コマンド・アイコン：プログラムの編集や実行などのシステム・コマンドを表わすアイコン

エクスターナル部はimage,label,shapeといった3つの属性から成る。imageはオブジェクトを象徴する絵シンボルを表わしており、labelはアイコンの名前を表わす。shapeはアイコンの枠の形を表わしており、typeに対応して図1に示すような形を持つ。

図3に、ソート・プログラムを表現するアイコンの例を示す。substanceはプログラムのオブジェクト・コードであり、conceptはソート、typeはプリミティブである。また、◈がimageであり、Q-Sortがlabelである。shapeは□である。

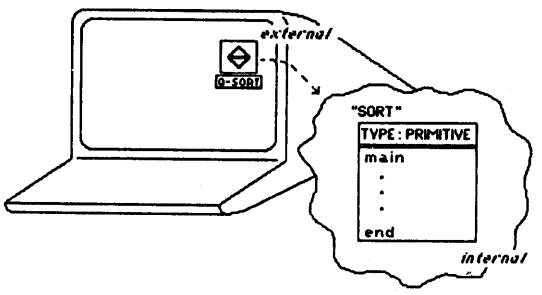

図3　アイコンの例

2．2　アイコンの管理

アイコンによるプログラミングやその実行を管理する際には次の9つの機能を提供することが要求される。

i）アイコンの作成：新しくアイコンをシステムに登録する機能

ii）アイコンの削除：不必要なアイコンを抹消する機能

iii）アイコンの操作：ディスプレイ上でアイコンの選択・移動などといった操作を行なう機能

iv）アイコンの実行：アイコンが示す機能を解釈／実行する機能

v）アイコンの分類・保存：アイコンをtypeやconceptなどで分類し、保存する機能

vi）アイコンの検索：v）で分類・保存されたアイコンを検索する機能

vii）アイコンの説明：あるアイコンに関する説明（アイコンが表している機能やオブジェクトなどについて）を行なう機能

viii）アイコンの表示：アイコンをディスプレイ上に表示する機能

ix）アイコンの接続：データの受け渡しを表現するためのアイコン間の接続関係を規定する機能

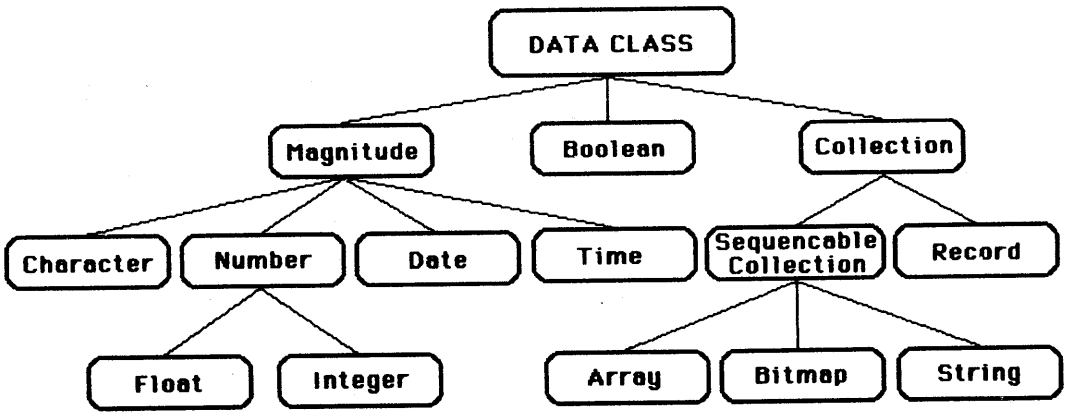

図2　データクラス階層

３．アイコニック・プログラミング

ＨＩ－ＶＩＳＵＡＬにおいては、アイコンとして表現されているオブジェクト（データやプログラム等）を画面上で直接操作することがプログラミングとなる。ここで、プログラムの作成はボトムアップあるいはトップダウンに行なえるようになっている。本章では、ＨＩ－ＶＩＳＵＡＬにおけるプログラミング方法について述べる。

３．１　インタラクティブ・プログラミング

アイコニック・プログラミングは大別するとｉ）アイコン間に接続関係を規定することによりアルゴリズムを視覚化してプログラムを記述するもの、ii）アイコンに一連の操作を施し、その操作の流れをプログラムとして保存するものの２つがある。後者は"Programming by Example"と呼ばれている[12]。

ＨＩ－ＶＩＳＵＡＬではｉ）の方法を基本に、ii）の特長である実行結果を見ながら対話的にプログラミングができるという利点も取り入れたプログラミングが可能となっている。

以下にプログラム作成手順を示す。

ｉ）ユーザはアイコン・メニューの中からマウスを用いて必要なアイコンを選択し、それをプログラミング・エリアの所望の位置に移動・配置する。

ii）配置されたアイコンが実行可能であればシステムはそのアイコンを直ちに実行し、実行結果を表示する。ここで、実行可能とは、入力のデータが全てそろった状態を指す。実行結果もまたアイコンとして管理される。

iii）ユーザは実行結果を参照し、再び必要なアイコンを選択・配置すると共に、アイコン間のデータの受け渡し関係を規定する。システムでは、接続されようとするアイコンの入力と出力のデータクラスが一致するか、あるいは、入力のデータクラスが出力のデータクラスのスーパークラスであれば、それらのアイコン間に矢印を設定する。データを受け取ったアイコンが実行可能であればii）と同様にアイコンの実行を行ない、その実行結果を表示する。

iv）上記のｉ）～iii）の操作をプログラムが完成するまで繰り返し行なう。

図４は、販売明細からどの品物がどれだけ売れたかという販売日報を作る事務処理プログラムの例である。

ＨＩ－ＶＩＳＵＡＬにおけるプログラミングの特長をまとめると次のようになる。

１）アイコンを選択・配置・接続することによって、プログラムを視覚的に作成することができる。
２）実行結果を見ながら対話的にプログラムの作成ができるので、デバッグが容易である。
３）プログラムはデータの流れを視覚化しており、処理の流れが把握しやすい。

３．２　トップダウン・プログラミング

インタラクティブ・プログラミングでは、アイコンの解釈・実行を行ないながらプログラムを作成していくので、

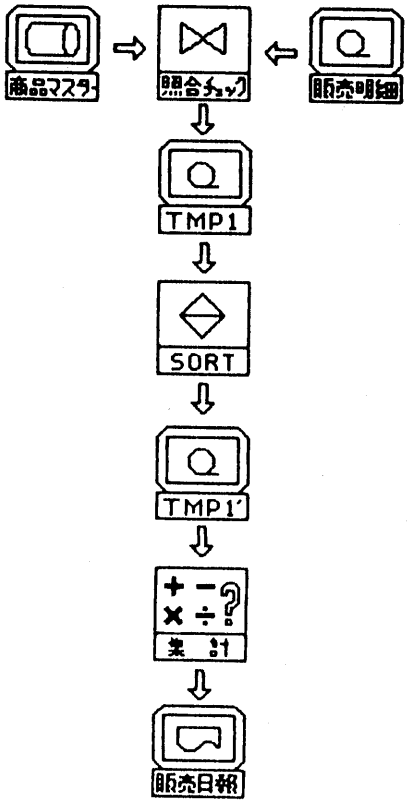

図４　アイコン・プログラムの例

ｉ）応答時間が長い、ii）詳細な仕様が決定されていないとプログラムの作成ができない、などといった欠点がある。そのため、ＨＩ－ＶＩＳＵＡＬでは、データ・アイコン（実データ）を使わず、代りにデータクラス・アイコンを用いることによって、プログラムの実行を見あわせたままプログラミングが行なえる。これによって、詳細なプログラム設計が終了していなくてもプログラム作成を進めていくことができる。すなわち、トップダウンによるプログラムの作成が可能となっている。

図５に例を示す。図の上側の太枠内のプログラムは、プログラムＰＲＧ１から出力されるＤＡＴＡ１というデータをプログラムＰＲＧ２に入力すると、ＤＡＴＡ２という結果が得られるというプログラムである。各プログラム・アイコン（ＰＲＧ１，ＰＲＧ２）には、モジュールの仕様を記述するサブウインドウが表示される。図５の下側の太枠内はＰＲＧ２を展開したものである。画面左上に展開中のプログラム・アイコンが表示され、その右にそのアイコンの仕様が表示される。それらの下側はプログラミング・エリアで、ここに後ほどＰＲＧ２が作成される。プログラミング・エリアにはあらかじめ入出力のデータクラスが記述されており、上位レベルとの整合性が保たれるようになっている。

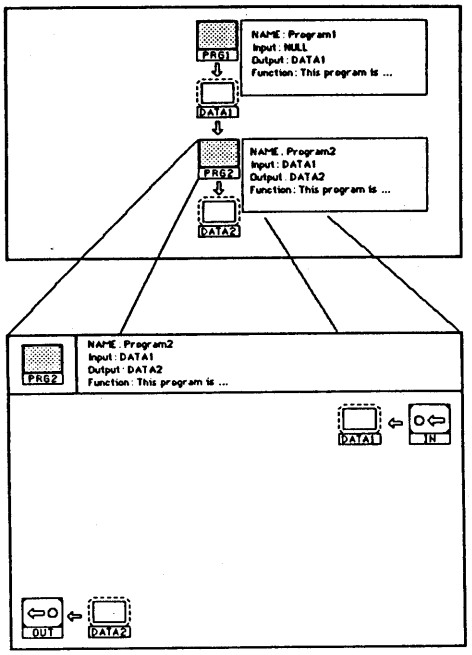

図5　トップダウン・プログラミングの例

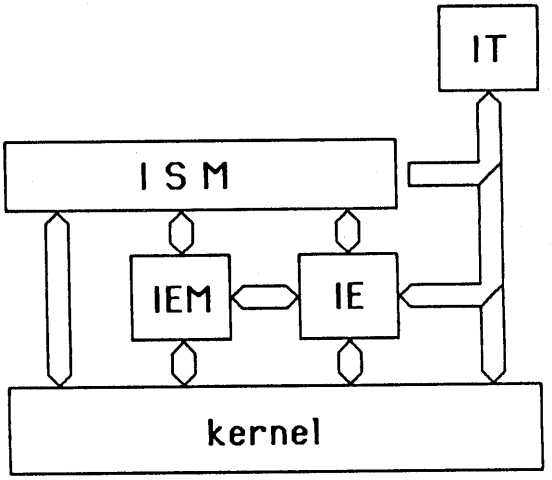

図6　システム構成

4．実験システム

4．1システム構成

システムは図6に示すように、5つのコンポーネントより構成される。

i）カーネル

カーネルはプロセス管理、デバイス・ハンドリング、ディスプレイ管理、データベース管理などといった、システムの基本処理の実行を行なう。

ii）推論エンジン（IE）

IEはIF−THEN形式によるルールベース・システムであり、カーネル内のデータベース管理システム（DBMS）の協力の下に、次の4つの機能を実現する。

①現在の状態とユーザの入力に基づいたユーザ・アクションの解釈

②アイコンのtypeとconceptに従った、アイコンの分類と管理

③ヘルプ要求に対する的確なユーザ支援情報の提供

④システム操作に関するナビゲーション

iii）アイコン実行管理部（IEM）

IEMはアイコン・プログラムの解釈・実行を管理する。解釈メカニズムはオブジェクト指向の概念を取り入れており、オペレータ・アイコン（プリミティブ／プログラム・アイコン）はデータ・アイコンに対するメソッドとして解釈される。

図7を用いてIEMにおけるプログラムの解釈手順を説明する。アイコン・プログラムは文献[11]と同様にディスクリプタを用いて管理さる。ここでは、アイコン・ディスクリプタ（ID）、ジョイント・ディスクリプタ（JD）、データクラス・ディスクリプタ（DD）の3つを用いて管理される。IDにはアイコンのインターナル／エクスターナル部に関する情報が記述されている。JDにはアイコン間の接続関係が記述されている。

以下にプログラムの解釈メカニズムを示す。

① IEMはJDに記述されているアイコンの接続関係に従ってデータをアイコンに受け渡す。

② データ（オブジェクト）を受け取ったオペレータ・アイコンはそのデータに対しオペレーション名をメッセージとして送る。

③ メッセージを受け取ったデータでは、そのデータが属するデータクラスにメッセージ・パターンと同じメソッド名を探す。注目しているデータクラス内に該当するメソッド名がなければそのスーパークラスにおいて、同様にメソッド名を探す。メソッド名がどのデータクラス内にも見つからなければエラー処理を行なう。

④ メソッドがプリミティブ・ファンクションであれば実行し、結果のデータを出力する。もしプログラムであれば、そのプログラムを展開して①からの操作を繰り返す。

⑤ 実行すべきアイコンがあれば①に戻ってそのアイコンを実行する。そうでなければ実行を終了する。

iv）アイコン・システム管理部（ISM）

ISMはシステムの中心をなすコンポーネントであり、他のコンポーネントの動作を管理する。IEとの協力の下にユーザ・アクションを解釈し、それに従って他のコンポーネントに指示を与える。

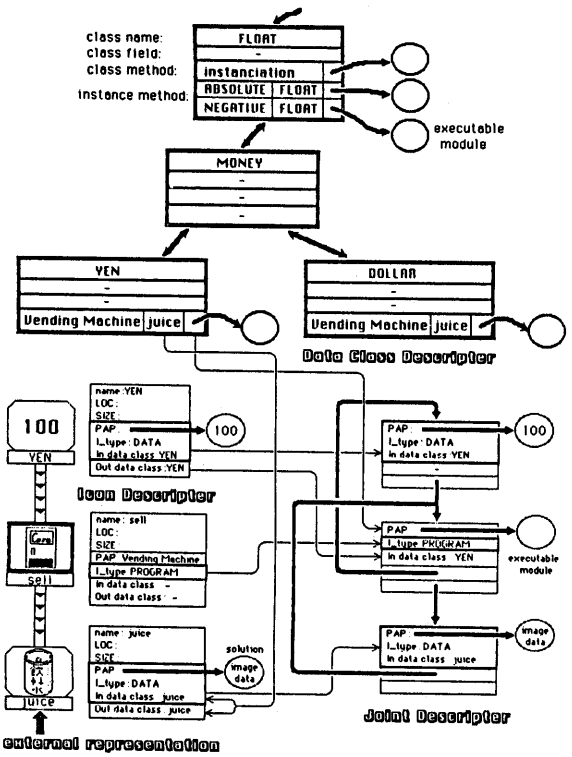

図7　アイコン・プログラムの解釈例

v）アイコン・ツール（ＩＴ）

アイコニック・プログラミングを支援するツールであり、次の５つがある。

①アイコン・イメージ・エディタ（ＩＩＥ）
　オブジェクトを象徴する絵シンボルを作成するためのイメージ・エディタ

②プリミティブ・バインダ（ＰＢ）
　プリミティブ・アイコンを作成するためのツール

③ブラウザ（ＢＲ）
　データクラスやオペレータクラス（後述）を検索するためのツール

④ヘルプ・システム（ＨＳ）
　ＩＥと共に動作して、ユーザの習熟度に応じた適切なヘルプ・メッセージを提供するためのサブシステム

⑤ユーザ・インタフェース・デザイナ（ＵＩＤ）
　ユーザが自分の好みにあったインタフェースを構築するためのツール

４．２　システム・オペレーション

すべてのシステム・オペレーションは、現在選択されているディスプレイ・オブジェクトに対するオペレーションとして解釈される。つまり、現在カーソルが指している表示物（アイコン、プログラミング・エリア、メニュ・エリア、ウインドウなど）に対するメッセージの発令であると解釈される。ここで、ＨＩ－ＶＩＳＵＡＬにおいては、ディスプレイ・オブジェクトは表示物とそれに対する操作をひとまとめとしたオブジェクトとして管理されており、オペレータクラスによって分類されている。ディスプレイ・オブジェクトに対する操作はオペレータクラスにメソッドとして記述される。これにより、オペレータクラスもデータクラスと同様に階層化され、メソッドの継承ができる。

オペレータクラスを利用して、現在選択されているディスプレイ・オブジェクトに対して行なうことができる操作をユーザに知らせることができる。すなわち、システム・オペレーションのナビゲーションを行なうことができる。例えば、図８に示すように、カーソルがプログラミング・エリア上にあると仮定する。ここでポップアップ・コマンドメニュを要求するアクションをユーザが行なった場合、システムは、プログラミング・エリアのオペレータクラスから発令可能なオペーションのリストを表示する。

また、一つのアプリケーション・プログラムを一つのウインドウとそれに対する操作としてオペレータクラスに記述することにより、既存のシステムをあたかも自分のシステムの一部であるかのように取り扱うことができる。すなわち、アプリケーションのコマンドをすべてコマンド・アイコンで表示するので、コマンド・アイコンを指すことによってアプリケーションのコマンドの実行を指示することができる。さらに、アプリケーションのコマンドに対するナビゲーションも可能である。

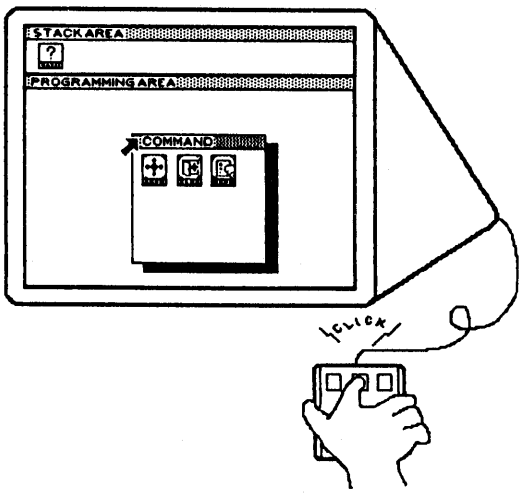

図８　システム操作のナビゲーション

5．おわりに

本論文では、ＨＩ－ＶＩＳＵＡＬの機能拡張について述べた。

拡張した機能は次の通りである。

（ｉ）プログラム開発やシステム操作のナビゲーション機能

（ii）オブジェクト指向の概念を取り入れたアイコン・プログラムならびにシステム・オペレーションの解釈メカニズム

（iii）ユーザ定義可能なユーザ・インタフェース

（iv）トップダウン・プログラミング機能

（v）既存システムの統合機能

今後の課題としては、

ｉ）推論エンジンの詳細設計

ii）アイコン・プログラムのコンパイル技法

iii）構造化データの要素に対する直接アクセス方法

などがあげられる。

謝辞

本研究を進めるにあたり、システムの設計およびインプリメントに際してご協力頂いた情報システム研究室の門田憲明氏（現在、日本データゼネラル）、田原義信氏、三谷和久氏、ならびに日頃ご討論頂く同研究室の諸氏に深く感謝する。

参考文献

[１] 平川、田中：＂視覚言語の動向＂、電子通信学会誌、Vol. 69、No. 12、pp.1256-1259、昭61-12.

[２] N. C. Shu: "FORMAL: A Forms-Oriented Visual-Directed Application Development System," IEEE Computer, Vol. 18, No. 8, pp.38-49, Aug. 1985.

[３] S. B. Yao, A. R. Hevner, Z. Shi, and D. Luo: "FORMANAGER: An Office Forms Management System," ACM Trans. on Office Information Systems, Vol. 2, No. 3, pp.235-262, Jul. 1984.

[４] E. P. Glinert and S. L. Tanimoto: "Pict: An Interactive Graphical Programming Environment," IEEE Computer, Vol. 17, No. 11, pp.7-25, Nov. 1984.

[５] L. Gould and W. Finzer: "Programming by Rehearsal," Byte, Vol. 9, No. 6, pp.187-210, Jun. 1984.

[６] S. P. Reiss: "PECAN: Program Development Systems that Support Multiple Views," IEEE Trans. on Software Engineering, Vol. SE-11, No. 3, pp.276-285, Mar. 1985.

[７] M. Moriconi and D. F. Hare: "PegaSys: A System for Graphical Explanation of Program Designs," Proc., ACM Symposium on Language Issues in Programming Environments, pp.148-160, Jun. 1985.

[８] R. V. Rubin, E. J. Golin, and S. P. Reiss: "ThinkPad: A Graphical System for Programming by Demonstration," IEEE Software, Vol. 2, No. 2, pp.73-79, Mar. 1985.

[９] 岩田、吉本、平川、田中、市川：＂アイコンプログラムの作成・実行環境の開発＂、情報処理学会第33回全国大会予稿、7F-4、昭61-10.

[10] N. Monden, Y. Yoshino, M. Hirakawa, M. Tanaka, and T. Ichikawa: "HI-VISUAL: A Language Supporting Visual Interaction in Programming," Proc., IEEE Workshop on Visual Languages, pp. 199-205, Dec. 1984.

[11] I. Yoshimoto, N. Monden, M. Hirakawa, M. Tanaka, and T. Ichikawa: "Interactive Iconic Programming Facility in HI-VISUAL," Proc., IEEE Workshop on Visual Languages, pp.34-41, Jun. 1986.

[12] R. V. Rubin: "Language Constructs for Programming by Example," Proc., ACM-SIGOIS Conference on Office Information Systems, pp.92-103, Oct. 1986.