グラフィクスとＣＡＤ 28－5
（１９８７・８・２０・２１）

ユーザインタフェース管理システムの構築と問題点（３次元ＣＡＤを例として）

川越恭二　　対比地幹雄
日本電気㈱

ユーザインタフェース管理システム（以下UIMS）の研究が活発化しているが、実際の応用システムの構築に使用したＵＩＭＳに関する報告は少ない。本稿では３次元ＣＡＤの研究開発の一環として開発したＵＩＭＳを説明し、このＵＩＭＳから得たＵＩＭＳ実現のための課題を説明する。

３次元ＣＡＤでＵＩＭＳを用いるためには入力や出力インタフェースだけでなく、対話の制御をいかにうまく行うかが重要である。この対話制御には１）コマンドのネスト、２）ＣＡＮＣＥＬ操作、３）メッセージ出力、４）コマンドマクロ化、５）多様な入力情報操作などを扱う必要がある。この問題を解決するために、対話言語を用いて対話制御を記述する方式を採用した。システムのインタフェース設計や応用システムプロトタイプとして使用する従来のＵＩＭＳに比べて、より応用システム（ＣＡＤ）向けでかつ利用者によるダイナミックなインタフェース操作が可能である。最後に、３次元ＣＡＤのＵＩＭＳ構築を通して、明かになった課題について説明する。

Implementation and Evaluation of A User Interface Management System For a Three-Dimensional CAD System

Kyoji KAWAGOE* and Mikio TSUIHIJI**

NEC Corporation

*　4-1-1, Miyazaki, Miyamae, Kawasaki, Kanagawa, 213 Japan
**　4-12-35, Shibaura, Minato-Ku, Tokyo 108, Japan

We developed a user interface management system (UIMS) for a three-dimensional CAD system in order to understand and evaluate problems involving user interface management system implementation. Our UIMS can uniformly perform not only input-output interface control functions, but also interaction control functions which are more important in CAD systems for efficient and user-friendly interactions.

For the interaction control, various kinds of interaction methods, including command nesting, cancel operation, macro-command defining and multi-media input-output information handling, have to be incorporated into the UIMS. To support these interaction methods, interaction control definition language is presented and dynamic interaction control is realized.

With this technique, efficient interaction manipulation by end-users suitable for large-scaled application systems, can easily be realized. This paper also describes unsolved problems on UIMSs obtained through experiences of this UIMS development.

1. はじめに

ユーザインタフェース管理システム（以下ＵＩＭＳ）の研究が活発化している[1-11]が、実際の応用システムの構築に使用したＵＩＭＳに関する研究開発報告は少ない。本報告では、既に実用化を行った３次元ＣＡＤを開発の一環として開発したＵＩＭＳについて説明する。

３次元ＣＡＤでＵＩＭＳを具体化するために検討しなければいけなかった項目とその解決策を説明するとともに、今回のＵＩＭＳの開発を通じて得た知見について記述する。

2. ＣＡＤにおけるＵＩＭＳ要件

まず、ＣＡＤでなぜＵＩＭＳが必要かについて説明する。

2.1 ＣＡＤの状況

３次元ＣＡＤのみならずどのようなＣＡＤでもその機能の多さは他のシステムには見られないであろう。例えば、点を作るという機能にしても、実際には、座標値を指定する方法、２つの直線の交点による方法、円と直線との交点による方法、線分の中点による方法など数多くの定義方法が提供されている。利用者にとっては点登録処理という単一機能でありながら、システム側では点を作成する多くの手順をあらかじめ用意しておかなければならない。

また、あらかじめ用意した作成機能は無駄な操作があったり、用意した作成機能では十分な作成が行えなかったときがある。このときには、ユーザ要求に答えるために、システム側の機能拡張を行わなければ利用者の望むシステムにはなりえない。

この２つの特性（豊富な機能さと多様な利用者要求）がＣＡＤシステムを大規模化する原因となっている。例えば、ＣＡＤシステムでは、１Ｍステップを越えることは驚くべきことではない。

このような状況を解決する手段として、利用者とシステムとのインタフェースおよび対話制御を一括してサポートするＵＩＭＳを用いることは自然であろう。なぜなら、ほとんどの機能は基本的な機能の組合せで実現することができ、この基本機能だけを応用プログラムとして扱い、その組合せを対話制御の一部分で行えば、システムの大規模化を防止することができる。また、利用者の新しい要求機能に対してはシステム機能を実現したのと同様の方式で基本機能の組合せで実現できる。さらに、インタフェースの変更についてはＵＩＭＳの持つ機能で自由に変更可能となる。

2.2 ＵＩＭＳ要件

実際に利用可能なＵＩＭＳは存在しないが、既に、研究レベルでの多くのＵＩＭＳの試作、開発が行われている[2]、[3]、[5]。これらのＵＩＭＳに共通の目標は、

1.一貫性のあるインタフェース
2.統一したインタフェース
3.応用プログラムとの独立性

の実現である。しかし、ＵＩＭＳをＣＡＤに適用するにあたってはその機能上、以下のような問題点が存在する。

1.多種類の図形種別：点、線以外に寸法線やスプラインのような図形の選択とその選択による対話フローの変更への対応
2.ダイナミックなインタフェース変更・追加：利用者自身によるインタフェースの変更が必要。
3.複雑な対話フロー：シーケンシャルな対話フローだけでなく繰り返し、条件分岐、キャンセル時の例外処理などの複雑な対話フローへの対応。

これまでＣＡＤを意識した唯一のＵＩＭＳはTiger[4]である。Tigerでは、対話記述言語TICCLを用いて対話制御内容を記述し、言語翻訳情報によって対話制御データに変換される。具体的な入力可能装置やアプリケーション処理はPASCALのプログラムで記述する。Tigerは処理と対話フローの分離、入出力装置との独立性などのＵＩＭＳの利点に加えてコマンドリピート、キャンセルの対話制御を統一して行う能力を持つものの、上記の1,2,3の問題は解決されていない。

2.3 ＵＩＭＳ機能

ＵＩＭＳをＣＡＤシステムで使用するには、上記の問題点を解決する必要である。具体的には、以下の機能を必要である。

１）コマンドのネスト。
２）ＵＮＤＯやＣＡＮＣＥＬ操作
３）適当な時点でのガイドメッセージ出力
４）ダイナミックなコマンドのマクロ化
５）多様な入力情報の解釈
６）複雑な対話フロー（繰り返し、条件分岐）
７）デフォルトとオプションパラメータ
８）入力可能コマンドの制限
９）階層化メニューコマンド
１０）コマンド履歴保存とその自動実行

１１）各種モードの切り替え、制御

３．　ＵＩＭＳ内部方式

前章で説明したＵＩＭＳ機能を実現するために開発したＵＩＭＳの方式を説明する。

3.1　全体構成

ＵＩＭＳの全体構成図を図１に示す。ＵＩＭＳの代表的構造である外部制御方式と内部制御方式[6]のうち、前者の構造を採用している。

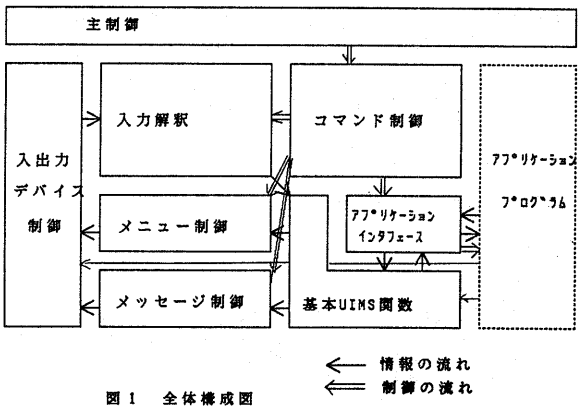

図１　全体構成図

図１における機能を説明する。

［入出力デバイス制御］：入出力装置の制御を行う。ＧＫＳなどの図形処理パッケージを含む。

［メッセージ制御］：ガイドメッセージやエラーメッセージの制御を行う。

［メニュー制御］：コマンド入力のためのメニューを制御を行う。

［入力解釈］：入出力デバイス制御を介して入力されたデータの内容を解釈して、対話制御可能な形式に変換する。カタログ言語は本機能の一部として実行される。

［主制御］：ＵＩＭＳやアプリケーションプログラムの実行、初期化や終了処理を制御する。

［コマンド制御］：ＵＩＭＳの本体部分であり、入力解釈の結果を判断し次の状態や処理を決定し、アプリケーション・インターフェースに制御を渡す。

［アプリケーションインタフェース］：アプリケーションプログラムへのデータ伝達や起動を行い、ＵＩＭＳとアプリケーションプログラムとを結合する。

［基本ＵＩＭＳ関数］：ＵＩＭＳとアプリケーション・インターフェースとの間の情報交換のための基本関数である。詳細は後述。

また、ＵＩＭＳのための言語とその位置づけを図２に示す。

図２における、言語を以下に説明する。

［対話制御言語（ソース）］：対話フローを記述するための言語。

［対話制御言語（オブジェクト）］：ＵＩＭＳで対話フローを制御するためのデータ。対話制御言語（ソース）を変換したもの。

［カタログ言語（ソース）］：利用者が対話フローやコマンドを変更、作成するための言語。対話記述言語（ソース）の記述内容により以降の対話言語に含めることができる。

［カタログ言語（オブジェクト）］：ＵＩＭＳで利用者作成コマンドや対話フローを制御するためのデータ。カタログ言語（ソース）を変換したもの。

［対話言語］：対話制御言語（ソース）によって記述されたコマンドとその対話フローからなり、利用者が業務で直接に使用する言語。

［システムＩ／Ｆ記述言語（ソース）］：画面レイアウトやメニュー構造など、対話フローを除いた利用者インタフェース情報を記述する言語。対話記述言語（ソース）の記述内容により先の対話言語に含めることができる。

［システムＩ／Ｆ記述言語（オブジェクト）］：ＵＩＭＳの入出力デバイス制御やメニュー制御などの使用するインタフェースデータ。システム記述言語（ソース）を変換したもの。

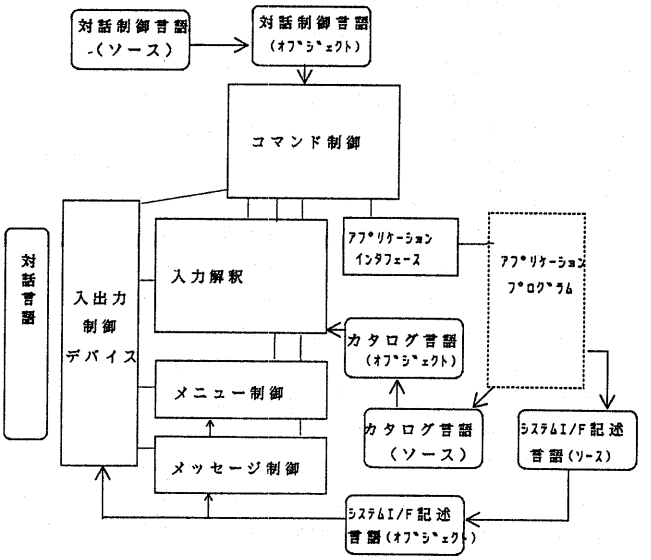

図２　ＵＩＭＳにおける言語

システム構築の際には、対話制御言語を用いてシステム開発者が対話フローを記述し、対話制御言語（オブジェクト）をシステム内の対話制御ファイルに蓄積する。

システム利用の際には、利用者が対話制御言語で記述された対話言語を使用してシステムと対話する。入力された対話言語は、対話制御ファイルの内容を参照し、適当なメッセージを出力したりメニューを制御しながら応用プログラムに制御を渡す。

利用者が、システムの対話フローを変更するには、直接に対話制御ファイルを編集する応用プログラムを使用するか、コマンド言語を使用して基本コマンドの組合せによって利用者コマンドを構築する。このコマンド言語はシステム運用中にダイナミックに行うことができる。

ＵＩＭＳを実現する上で解決すべき問題は、主に１）アプリケーションとの結合、２）入出力の方法、３）対話制御フローの記述である。以下に、これらの問題を解決するために使用した方式を説明する。

3.2 アプリケーションとの結合方式

ＵＩＭＳからアプリケーションへ制御を渡す際の問題は、以下に情報交換を行うかである。すなわち、制御を渡すときに入力データを編集してアプリケーションへ渡す方法と、制御がＵＩＭＳに戻されたときに出力データを編集して、画面に表示したり次のアプリケーションにそのデータを渡す方法とを明確にする必要がある。

ＵＩＭＳの中には、利用者が自由にアプリケーションプログラムとの結合を記述するプログラムを作成するため上記の問題は発生しないが、コマンドの起動だけでなく対話フローの詳細をＵＩＭＳで制御するときには問題となる。

この問題を解決するために以下の方針を採用した。

１）アプリケーション・インターフェース・プログラムを機能（コマンド）ごとに用意する。そのインタフェース・プログラムの構造は、ＵＩＭＳから制御可能なものとする。

２）上記プログラムへ入力データを渡すために、専用の関数を提供し、制御をこのプログラムに渡すときにはデータは渡さない。

３）メッセージ以外の出力データについても専用の関数を提供し、この関数を介して画面にデータを表示する。なお、大量の図形を表示するときには直接に入出力デバイスを使用する。

４）操作ガイドなどのメッセージについては、メッセージ文ではなく識別子によって専用の関数で表示させる。

図３に、この方法を示す。図４にアプリケーション・インターフェース・プログラムの構造、記述例を示す。

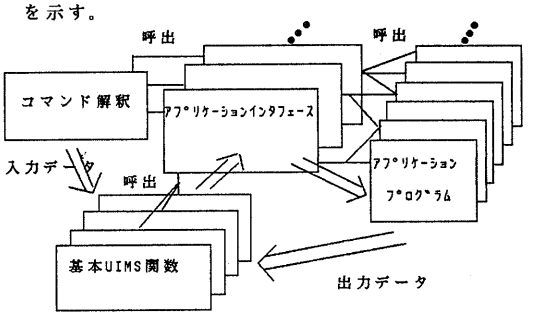

図３　入出力データ伝達

```
XXXXXX(CMND,MODE,SNO,ACT,OPT,RETURN,ERROR)
  CMND:コマンド番号、MODE:コマンド入力後の呼出回数,SNO:文番号、
  ACT:アクション番号（SNO、ACTともに対話制御言語で記述されている）
  OPT:オプション記号の入力有無、RETURN:戻り時の文番号、
  ERROR:コマンド制御へのエラー通知

ENTRY:
    RETURN=0
    ERROR=0
  CASE:
    OF: SNO=0
          コマンドの初期化
    OF: SNO=10
          CASE:
            OF: ACT=10 OR 11
                原点を始点に設定
            OF: ACT=12 OR 15
                座標値を始点に設定
            OF: ACT=13
                図形要素の座標値を始点に設定
            OF: ACT=14
                図形要素名からその図形要素の座標値を始点に設定
            OTHER:
                エラー処理
          END:
    OF: SNO=20
          CASE:
            ||
            OF: ACT=20
                始点＋増分値を終点に設定
            ||
          END:
    ||
    OF: ACT=-1 ;CANCEL入力
          CASE:
            OF: SNO=10
                状態１０でのキャンセル処理（必要なら）
            OF: SNO=20
                状態２０でのキャンセル処理（必要なら）
            ||
          END:
    ||
    OF: ACT=-2  ;BREAK入力
                ブレーク処理（必要なら）
    OTHER:
          エラー処理
    END:
EXIT:
```

図４　アプリケーション・インタフェースの構造

このような方法を採用した理由は、1）将来的にはインターフェース・プログラムを対話制御言語から自動生成することができること、2）個数、形式ともかなり変化する入出力データを一括で渡すことは不可能なこと、3）専用関数でデータを伝達するため移植、保守、拡張が容易であることなどである。

専用関数の一覧を表1に示す

表1　基本UIMS関数（一部）

機能名	機能説明	パラメータ
入力図形取り込み	ピッキングや名前指定で選択された図形要素を参照する。	図形識別子、図形名、ビュー番号、ピッキング点の座標値
座標値取り込み	座標値入力デバイスにより入力された座標値を参照する。	座標値、ビュー番号、視線ベクトル
座標値入力	文字列入力デバイスより入力された座標値を参照する。	座標値
文字列取り込み	文字列入力デバイスより入力された文字列を参照する。	文字列個数、文字数と文字列の集合
実数値入力	文字列入力デバイスより入力された実数値を参照する。	実数個数、実数値の集合
バリュエータ入力	バリュエータから入力された変位量を参照する。	量子ファクタ、変位量
モード参照	各種モードの参照、設定	設定・参照区別、モード値
メッセージ出力	強制メッセージの出力を行う。	メッセージクラス、文字列
画面更新	図形表示後のいくつかの処理の結果、作成された図形を表示する。	ビュー番号

3.3　入出力との結合方法

ＣＡＤシステムでは、大量の図形を表示するとともに、タブレット、マウス、キーボードなどの多種類の入力装置を使用する。このため、ＧＫＳなどの図形処理パッケージによって、独立性の向上は重要なことである。しかし、Tigerにおいても、図形処理パッケージでの装置の分類がＵＩＭＳでの対話制御に必要な分類と異なると指摘している[4]ように、本ＵＩＭＳでもこのことが重要な問題である。

このため、入力デバイスを通常のものでなく、対話制御を意識した以下のものに変更した。

1）座標値入力デバイス：タブレットやダイヤルをなど用いてカーソル位置の（2次元、3次元）座標値を読み取る。また、キーボードを用いて座標値を入力する。

2）図形要素選択デバイス：マウスやタブレットなどを使用して、画面に表示されている図形要素のピッキングを行う。

3）バリュエータ：ＣＯＲＥやＧＫＳのバリュエータと同じ。数値を入力するのに使用する。

4）任意文字列入力デバイス：キーボードやあらかじめ文字列が設定されたファンクションキーなどを使用して、文字列を入力する。

さらに、入力データの解釈の際に、選択した図形要素がどのようなものかを知る必要があるし、選択したときのカーソルの座標値も対話制御で使用している。このため、入力データは、単に利用者が入力したデータ以外のデータも必要となる。以下に、その形式を示す。

入力データ型、パラメータ並び

（例）図形選択、図形要素タイプ、名前、セグメントタイプ、セグメント名、ビュー、座標値（ワールド座標系）、座標値（正規化デバイス座標系）

出力データの内容で対話フローが変わることは、エラー発生のとき以外にない。このため、特に考慮すべきことはない。

3.4　対話制御言語

対話制御言語の概略を説明する。

対話制御言語は対話言語を記述するための言語であり、以下の文から構成される。

1）普通文：対話フローを構成する基本対話（メッセージ出力、入力解釈、処理起動）を記述する文。

2）ループ文：対話フローのなかで繰返し処理の開始と終了を記述する文。

3）終了文：個々のコマンドの対話フローの記述の終わりを示す文。

特定のコマンドに関する対話フローは、上の文の組合せで記述される。これを一つの文章として扱う。なお、メニュー階層やファンクションキーの割当はシステムＩ／Ｆ記述言語で定義される。

普通文、ループ文の形式と例を図5に示す。

この対話制御言語で記述可能な文章を以下で説明する。

1）対話言語の文章は複数の文より構成される。

2）単純文章と複合文章：コマンド入力後、データやオプション入力なしにアプリケーションプログラムを起動するものを単純文章という。それ以外はすべて複合文章である。

3）文は、一個以上の記号の並びから構成される。

4）記号：記号は、システムであらかじめ定義された文字列や利用者が入力したデータなどをいう。

5）入力可能記号：ある状況で入力可能な記号を入力可能（エネーブル、enable）記号という。入

力可能記号は記号を入力するにつれて絶えず変化する。

6）ループ処理：特定の処理を指定記号入力まで繰り返す処理。ループ処理の範囲は、ループ開始点からループ終了点までの範囲である。ループ処理は、必ず発生するときと指定記号入力で起動されるときがある。ループの開始は必ず開始点で行い、ループの脱出は必ず終了点で行うものとする。ループのループは認めない。ループの回数は任意である。

7）オプション：コマンド入力ののち、対話フローにそった制御が行われる最初の段階で、必要ならば入力可能な記号をオプション記号という。既定値や標準処理を変更したいときに使用できる。オプション記号は入力可能記号であれば任意個入力でき、その順序に制限はない。オプション記号を対話フローの途中には記述できない。

8）ネスト：文のなかに他の文章を埋め込むことをネストという。ネストされる文章はコマンドの投入で行われる。その文章の終了後は、ネストした文の次に制御が渡る。

9）キャンセル（CANCEL）とブレーク（BREAK）：キャンセルは直前に処理された記号をキャンセルするＵＮＤＯ機能であり、特殊なＣＡＮＣＥＬ記号が用意されている。一方、ブレークは文章全体を強制的に終了させる。特殊なＢＲＥＡＫ記号が用意されている。

10）分岐：特定の時点での分岐は、入力されたデータと入力可能記号との比較によって行われる。

11）選択：特定の時点で入力可能記号が複数個存在し異なるアプリケーションプログラムが処理されるが以降の対話フローは変更されないことを選択という。

形式

(1) 普通文

```
文番号:STD:
   NMSG=m/MSG=M1…Mm/
   NOPT=i/NGRP=j/
    ACT=A1､WORD=P1…､CTL=NONE OR LOOP/
                        :
                        :
   NWORD=k/NGRP=d/
    ACT=B1､WORD=W1…､GOTO=11,L2､CTL=NONE OR LOOP/
                        :
                        :                    ;
```

(2) ループ文

```
文番号:LSTA:
   TYPE= YES OR NO
   NMSG=m/MSG=M1…Mm/
   NOPT=i/NGRP=j/
    ACT=A1､WORD=P1…､CTL=NONE OR LOOP/
                        :
                        :
   NOTHER=n/NGRP=g/
    ACT=C1､WORD=t1…､CTL=NONE OR LOOP/
                        :
                        :
   NWORD=k/NGRP=d/
    ACT=B1､WORD=W1…､GOTO=11,L2､CTL=NONE OR LOOP/
                        :
                        :                    ;
文番号:LEND:GOTO=1
   NMSG=m/MSG=M1…Mm/
   NOPT=i/NGRP=j/
    ACT=A1､WORD=P1…､CTL=NONE OR LOOP/
                        :
                        :
   NWORD=k/NGRP=d/
    ACT=B1､WORD=W1…､GOTO=11,L2､CTL=NONE OR LOOP/
                        :
記述例                  :                    ;
```

(1)普通文

```
 10:STD: NMSG=1/MSG=1000/
         NOPT=0/
         NWORD=6/NGRP=6/
            ACT= 10, WORD=INC, GOTO 20/
            ACT= 11, WORD=ORG, GOTO 30/
            ACT= 12, WORD=IND, GOTO 30/
            ACT= 13, WORD=HPT, GOTO 30/
            ACT= 14, WORD=NPT, GOTO 30/
            ACT= 15, WORD=XYZ, GOTO 30 ;
```

(2)ループ文

```
  40:LSTA:TYPE=NO
          NMSG=1/MSG=1002/
          NOPT=2/NGRP=1/
             ACT= 40,WORD=MOVE,COPY/
          NOTHER=1/NGRP=1/
             ACT=41,WORD=KB,GOTO=END/
          NWORD=2/NGRP=2/
             ACT=42,WORD=HALL,GOTO=50/
             ACT=43,WORD=NALL,GOTO=50;
  50:LEND:GOTO=40/
          NMSG=1/MSG=1004/
          NOPT=0/
          NWORD=1/NGRP=1/
             ACT=50,WORD=YN,GOTO=60;
```

図５　対話制御言語の形式と記述例

3.5 カタログ言語

カタログ言語は、利用者がダイナミックに対話操作を作成・変更するための言語であり、基本コマンドの組合せを定義するものである。

1）基本コマンド：基本コマンドとはあらかじめシステムで用意された最小限のコマンドをいう。例えば、形状のSAVE,LOAD,MOVE,ROTATEなどである。

2）組合せ：複数個の基本コマンドを順序列、選択、条件分岐、繰返しによって組み合わせて新しいコマンドの生成を行う。この組合せには、対話制御言語と同一の機能、形式を使用することができる。

3）基本関数：コマンドを定義するために必要な計算、メッセージ出力、入力要求などを記述する。

対話制御言語での基本関数は、アプリケーションインタフェースプログラムで扱えるために、ＵＩＭＳとアプリケーションプログラムのデータ伝達に関するもののみであるが、カタログ言語では、動的な生成を実現するために、それ以外に代入関数、座標値抽出・合成関数などが用意されている。

4）変数：同じ図形要素や値を使用したりするときにその要素や値を変数とすることで、必要なときに参照することができる。変数には、利用者が行ったデータ入力、基本コマンドの実行によって生成した図形要素や値、繰返し回数、代入や計算で生成した値などがある。

カタログ言語の例を図６に示す。

```
CBEGIN: NAME=xxxxxxxx MSG='GENERATE BOX'
  SET ROT3=1.7320508
  MSG 'INPUT POSITION'
  SET L=1.72
  SET PAR=NULL
  PNT1=?POS
  PT1X=X(PNT1)
  PT1Y=Y(PNT1)
  PT1Z=Z(PNT1)
     :
     :
  LOOP: PAR=?YN
    MSG 'INPUT POSITION OR YN'
    CASE:
     OF:?POS
       PNT2=?POS
       LINE PTPT %HPT=PNT1  %HPT=PNT2 /LNO1
       SET PNT1 = PNT2
     OF:?YN
       PAR=?YN
     OF:?CANCEL
       DELETE LINE LNO1
       LNO1=NULL
  LEND:
CEND:
```

図６　カタログ言語記述例

４．　考察

ここでは、３章で説明したＵＩＭＳについての考察を行う。

4.1 従来のＵＩＭＳとの違い

従来のＵＩＭＳは、エディタ、ファイル管理ｼｽﾃﾑやｸﾞﾗﾌｯｸｽｼｽﾃﾑなど対話機能としては比較的小規模のシステムを第一の対象としたものが多い〔2,3,10〕。実際の大規模ＣＡＤシステムのためのＵＩＭＳを実現するために、複雑な対話フロー、ダイナミックな対話記述、多様な入出力情報操作などを解決する必要があった。また、従来のＵＩＭＳで議論されている入出力機能〔2〕については画面レイアウトやメニュー構成などは技術的問題もなく容易に実現することができた。さらに、インタフェース構築のためのツールキットの充実というアプローチ〔3,7〕と異なり、実行部での対話制御機構を重視している。これは、ＣＡＤシステムという大規模システムを対象としており新規システムの構築よりもシステム機能の拡張という作業が多いためである。構造的には、従来のＵＩＭＳの対話システム生成用プリプロセッサ〔7,8〕や対話制御ライブラリー〔3〕という方式よりもより完全なＵＩＭＳである。

4.2　反省点

本ＵＩＭＳの開発によりＵＩＭＳ応用システム（すなわち３次元ＣＡＤ）の開発期間の短縮、インタフェースの統一化、インタフェース保守の用意さが実現された。しかし、以下のような問題があることが判明した。

1）大規模化：インタフェースの統一処理化の実現のために全ての対話制御に関するメソッドを一つのＵＩＭＳ内部に保持するため、ＵＩＭＳ自体が大規模になり、３次元ＣＡＤシステムの大規模化を促進してしまった点。また、大規模化に伴い処理効率の悪化が避けられない点。

2）標準化：インタフェースの統一化の反動として、柔軟なインタフェースを記述できなくなった点。すなわち、統一化はインタフェースの標準化を意味し、部分的で特殊であるが効果的なインタフェースを実現することが困難となった点。

3）独立性：ある程度のアプリケーションプログラムとの独立性は実現されたが、アプロケーションインタフェース部の存在によって対話制御記述の一貫性がやや犠牲になった点。また、入出力デバイスとの独立性は向上したが、低機能端末を使用するときに高機能メニューを実現するに伴う開発量が増加する点。また、新しい図形要素を加えたときにＵＩＭＳ自体の改造が必要な点。

4.3　課題

4.2の反省点や現在の環境を考えれば今後のＵＩＭＳは以下の点を解決する必要がある。

1）マルチウインドウ化やアイコン：最近のウィンドウシステムやアイコンシステムとの関係を明確化する必要がある。図１の入出力デバイス制御に含めることが可能である。

2）入力情報クラスと対話制御との非独立性：対話フローの制御に必要な情報のクラスをＵＩＭＳ

で記述可能な機能（対話制御メタ記述）が必要。

3）アプリケーションプログラムへの情報伝達方式や独立性：ＵＩＭＳの外部制御方式におけるアプリケーションプログラムへの情報伝達の方法。図１のアプリケーションインタフェースの自動生成が必要。

4）大規模化と性能劣化：ＵＩＭＳの大規模化を避ける工夫と、制御部分のオーバヘッドの削減。

5）汎用性と柔軟性のトレードオフ：インタフェース統一化による柔軟性の犠牲の回避。対話記述の制約をはずす必要があるとともに、対話記述モデルの確立が必要。。

6）対話記述言語（制御・カタログ）の操作性の重要性：対話を記述する言語のわかりやすさ、効率、適応化がＵＩＭＳでは重要である。特に利用者が記述する場合にプログラミングを不要にする必要がある。

7）知的化の必要性：インタフェースへの利用者要求は限りないものであるためＵＩＭＳの目標も際限なく高度になっていく。しかし、対話記述の拡張性、互換性を保証する必要がある。このためには、ＵＩＭＳに推論機能（演繹、帰納）が有効であろうがその利用方法の検討が必要。

5．　おわりに

本稿では、３次元ＣＡＤシステムの開発の一環として開発したＵＩＭＳについて報告した。

開発したＵＩＭＳは、従来のＵＩＭＳに欠けている複雑な対話フローの記述が可能なこと、利用者自身でダイナミックに対話制御を記述可能なことなどを特徴としており、本稿ではその内部方式について説明した。

本ＵＩＭＳは５年前の２次元ＣＡＤシステムの開発の一環として開発したＵＩＭＳを原型としたものであり、３次元ＣＡＤ以外にも２次元ＣＡＤでも利用可能である。また、ＵＩＭＳはシステムに埋め込まれて約３年前から実際の利用者へ提供され使用されている。

本ＵＩＭＳの開発経験を元に、より利用者に対話操作を適応可能なシステムと、そのためのＵＩＭＳの研究を現在行っている〔12,13〕。

［謝辞］

本稿で記述したＵＩＭＳは３次元ＣＡＤの開発メンバーによって開発されたものであり謝意を表す。

参考文献

〔1〕ACM SIGGRAPH Workshop on siftware Tools for User Interface Management, ACM Computer Graphics, Vol. 21, No. 2, pp.71-147, 1987

〔2〕Hayes, P. J. et al.: Graceful Interaction through the COUSIN command Interface, Int. J. of Man-Machine Studies 19. 3, pp. 285-305,1983

〔3〕Schulert, A. et al.: ADM- A Dialog Manager, Proc. of ACM CHI'85, pp.177-183, April, 1985

〔4〕Kasik, D.: A User Interface Management System, Computer Graphics, Vol. 16, no.3, pp.99-106, 1982

〔5〕Wong, P.C.S. et al. : Flair- User Interface Dialog Design Tool, Computer Graphics, Vol 16 no. 3, pp.87-98, 1982

〔6〕Workshop Summary: Graphical Input Interaction Technique, CVomputer Graphics, pp.5-30,January,1983

〔7〕Buxton, W. et al.: Towards a Comprehensive User Interface management System, Computer Graphics, 17, 3, pp.35-42, 1983

〔8〕Olsen, D. R. et al.: SYNGRAPH: A Graphical User Interface Generator, Computer Graphics, 17, Npo. 3, pp.43-50, 1983

〔9〕Olsen, D. R. et al.: A Context for User Interface Management, IEEE CG&A, pp.33-41, 1984

〔10〕今宮他：柔軟なユーザインタフェースの作成、グラフィックスとＣＡＤシンポジウム,pp.1-8,1985

〔11〕守屋他：ﾕｰｻﾞｲﾝﾀﾌｪｰｽ管理システムの方式とその試作、情報処理学会グラフィクスとＣＡＤ研究会資料、25-2、1987

〔12〕川越他：適応型対話インタフェースを持つ知的システムとその応用、第２回ヒューマンインタフェースシンポジウム、pp.25-32、1986

〔13〕浜川他：適応型対話インタフェースを持つ空間プランニングシステム、情処全国大会３３回2M-5、1986

討論

5．ユーザインタフェース管理システムの構築と問題点　　川越（日電）

坂下：記述言語を３つ用いていますが、デバッグが大変ではないでしょうか。

川越：実際に開発者が使う言語は対話制御言語とアプリケーションインタフェースのプログラムで、これらを同時に作成するのが難しさのポイントです。カタログ言語とシステムＩ／Ｆ記述言語はこれらの言語とは独立のもので、単独でデバッグできます。対話制御言語からアプリケーションインタフェースのプログラムを自動生成できれば良いのですが、まだそこまでは行っていません。

守屋：課題のところに関して、システムの大規模化と性能劣化についての具体的数値があれば教えてください。

川越：数値はあげられませんが、確かにＵＩＭＳを使わないものと比べると遅いです。大規模化は制御フローの複雑化に起因しています。

守屋：全体構成図（図１）を見ると、アプリケーションに到達するまでにいろいろなパスを通っていますが、速度に問題はありませんか。

川越：これは既に製品になっていまして、速度も実用的な範囲に収まって、かなり使い勝手は良くなっています。

福井：ユーザが変更できるのはどの部分ですか。

川越：コマンドの組合せ、つまり対話制御のフローと、システムＩ／Ｆ記述言語によりメニューや画面のレイアウトを変えられます。
対話制御言語は一般ユーザには解放していません（リンクし直さなければならないので）。

守屋：基本ＵＩＭＳ関数（表１）はどれぐらい用意されているのですか。

川越：ここに挙げただけで２０ぐらい、さらにいろいろなものが用意されています。

坂下：それぞれのアプリケーションごとにＵＩＭＳが組み込まれるのですか。

川越：全体で１つの実行形式のモジュールです。したがって大きなシステムになりますが、複数が同時に動くようなことはあまりないので問題はありません。

村上：これを用いることによって、アプリケーションを作るのにどれぐらい開発期間が短縮されたか教えてください。

川越：比較は難しいですが、１回目の開発ではＵＩＭＳ自体の開発も含まれるため全体としては同じぐらいだと思います。２回目以降は短縮されます。

守屋：基本ＵＩＭＳ関数が保持するデータのモデルはどういうものですか。

川越：入力されたものに対して持つ情報は入力種別と座標値です。出力に関しては、変更した情報を保持する場合には図形の識別子などを持っています。