Eulerオペレーションに基づく形状記述システムの試作

小島俊雄（機械技術研究所），三末和男，榎本進（東京理科大学），
井上久仁子，関口博（機械技術研究所）

1．まえがき

CAD・CAMシステムなど，機械工業の生産に関する各種ソフトウェア製品（本文では生産ソフトウェアと呼ぶ）が次々と生まれている。これらのシステムには，コンピュータ・グラフィックスや，ソリッド・モデリング・システムのような共通的な要素もあるが，全体的には個々に設計・開発・保守されている。また，これらのシステムの用途も多方面に拡がり，一つの汎用システムでユーザの多様な要求を満たすことはよりむずかしくなってきている。

そこで，システムをオープン・エンディッドにすることにより，ユーザ向きのシステムを構成することを試みた。以下，実験的な形状記述システムのソフトウェア・ツールを用いた構成例について，システムの内容及びその評価を述べる。

2．基本概念

まえがきに述べた，各々のユーザが満足いくシステムを構成することが困難である点の一つに，一般にユーザは独自のデータ構造，コマンド体系を用いて，自分流の仕事をすることを希望するし，システムもそれに応じるべきであるということがある。つまり，システムはそのデータ構造やコマンド体系にユーザの意見をとり入れるべきである。

もちろん，ユーザ自身が自分に合ったシステムを作る，という方式も考えられるが，現状では，このためにはユーザはアプリケーションに精通するとともに，高度のプログラマーでもなければならない。

そこで，本文で示すシステムは，基本的なデータ構造と基本操作を基本システムとしてあらかじめ用意し，ユーザがその基本システムの上にユーザ独自の操作・コマンドを作成する方式を提案するものである。この場合，ユーザは現状ほど高度のプログラマーである必要はなく，自分に合った操作・コマンドを作成するための手順のみを知っていれば，自分向きのシステムを作成できることになる。

3．システムの試作

3．1　システムの構成

2.で述べた概念に沿って，ユーザが自分向きの形状記述法を利用できる実験用の形状記述システムを作成した。この形状記述システムはユーザがキーボードからコマンドを入力し，その結果をディスプレイ装置上で確認する対話型のシステムである。

基本データ構造としては，ウィングド・エッジ・データ構造を，基本操作としては，オイラー・オペレーションをそれぞれ用いている。その試作システムの構

成をFig.1に示す。

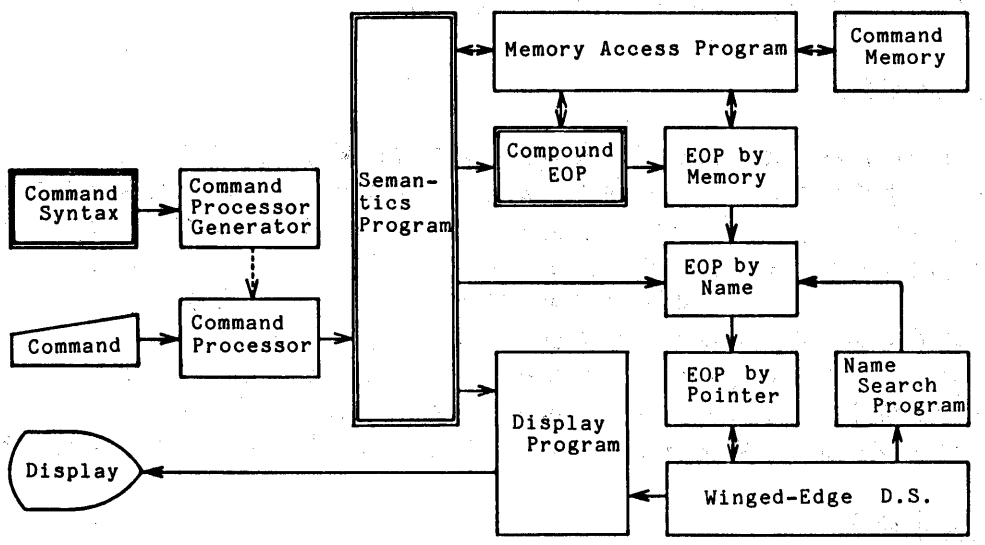

Fig.1　試作システムのソフトウェア構成

本システムのプログラムの大部分はPascalで記述されており，Fig.1の各ブロックに対応する手続きの大きさは，ソース・プログラムで以下のようになる。

手　続　き	（行）
Semantics	1500
Display	1700
Compound EOP	550
EOP/M & Memory Access	250
EOP/N & Name Search	200
EOP/P	1300
Main & Others	400
Command Processor Generator	1500
計	7400（行）

オイラー・オペレーションには，Fig.2に示すように，データ構造の各図形要素（面，辺，頂点など）の指定の仕方によって3種類の手続きが存在する．各図形要素はPascalのrecord型で宣言されていて，そのrecordの成分として各図形要素のデータの他に，integer型の名前を持っている。

基本的には，各図形要素はそれに対応するrecordを指すポインターによって指定される。このようなオイラー・オペレーションの手続きを，ポインターによるオイラー・オペレーション（EOP/P）と呼ぶ。

しかし，ユーザがコマンドでポインターを直接あつかうことはできないので，

図形要素の名前を使って指定する。このようなオイラー・オペレーションの手続きを，名前によるオイラー・オペレーション（ＥＯＰ／Ｎ）と呼ぶ。

また後出の複合操作のために，一つの手続きを特定の図形要素に対してだけでなく，不特定の図形要素に対して用いることができなければならない。このため，変数的な役割りをさせるメモリーを用いて，その番地で要素を指定する手続きを加えた。このようなオイラー・オペレーションの手続きを，メモリーによるオイラー・オペレーション（ＥＯＰ／Ｍ）と呼ぶ。

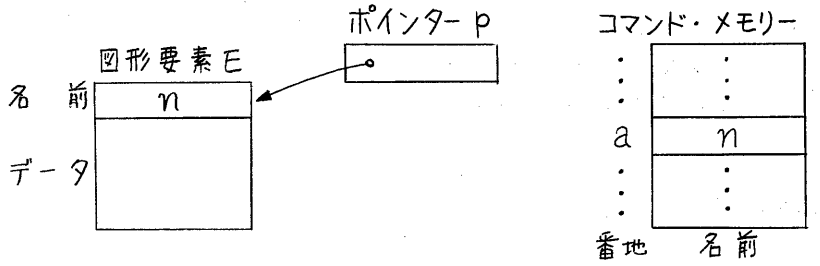

Fig.2　図形要素の３通りの指定の仕方
図形要素Ｅは，ポインター Ｐ（EOP/P），名前 n（EOP/N），番地 a（EOP/M）のどれでも指定可能

ＥＯＰ／Ｍで使うメモリーはコマンド・メモリーと呼ばれ，各図形要素ごとに用意されている。また図形要素の名前と別に，座標用のメモリーも用意されている。

Fig.1で二重線で囲んで示した部分は，ユーザが独自の操作・コマンドを作成する場合に，自由に書きかえることのできる部分である。

３．２　基本システム

基本システムはこの形状記述システムの核となるもので，その操作はオイラーオペレーションのみで，あらかじめシステムで用意されている。
たとえば，Fig.3のように頂点と辺を作るコマンドＭＥＶは，

$$\mathrm{MEV}\left\{\begin{array}{l} V/n_1 \quad (x,y,z) \\ (x,y,z) \quad V/n_1 \end{array}\right\}\left[E/n_2\left(\left\{\begin{array}{l} CW \\ CCW \end{array}\right\}\right)\right]$$

{…}：選択，[…]：省略可

のように定義されている。この操作は頂点n_1と新しい頂点（ｘ，ｙ，ｚ）の間に辺n_2からCW（時計回り）またはCCW（反時計回り）の方向に新しく辺をつくる。

コマンドは，MBFLV，KBFLV，MEV，KEV，MEF，KEF，KEMR，MEKR，KFMRH，MFKRH．のオイラー・オペレーションの他に，入出力制御を行なう，CONSOLE，FILE，DISPLAY，DSSET 等がある。これらのコマンドは全て，あらかじめ定義されているが，後で述べるユーザ定義の方法を用いて，独自のシ

ンタックスによって再定義することも可能である。

基本システムでは，オイラー・オペレーションのみを，また後出のその応用についても，オイラー・オペレーションのみを基にしているので，ユーザはそれより細かいことを考慮する必要はない。つまり，データ構造として，たとえば，具体的にどのようなデータ構造が用いられているか，またそれがどのように用いられているかなど知る必要はない。

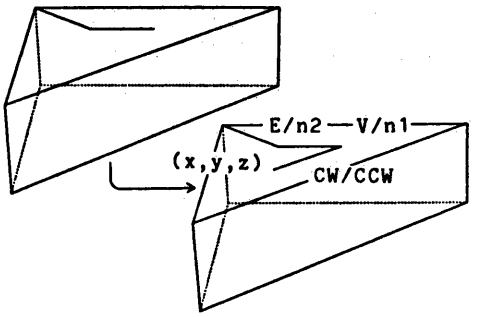

Fig.3 Make Edge and Vertex

３．３ 複合操作

ユーザは基本システムの操作をいくつか組み合わせて，複合操作とすることができる。この複合操作を用いることによって，基本操作では多数のステップが必要な複雑な形状に関して，より少ないステップで記述できるようになる。

たとえば，正N角柱を作るためには，MBFLVを１回，MEVを２N-１回，MEFをN+1回適用すればよい。そこで，これを一つの操作にすることにより，１コマンドで正N角柱を作ることができる。

３．４ 複合操作・コマンドの作成

複合操作をユーザが定義する場合，実際にユーザが行なう作業は以下の様になる。（Fig.4参照）

① 複合操作の基本操作による構成

複合操作はPascalの一つの手続きとして記述し，その中でＥＯＰ／Ｍに対応する手続きを呼んで基本操作の機能を利用する。その他の組み込み関数，制御文等のPascalの機能も全て利用できる。

実例として，正多角柱の場合を考えてみる。底面が半径Rの円に内接する高さＺの正N角柱を作る操作で，新しくできた立体の名前をコマンド・メモリーのRET番地へ返し，BASE番地以降をこの操作のワーク・エリアとして一時的に用いるとすると，実際のPascalの手続きはFig.5の様

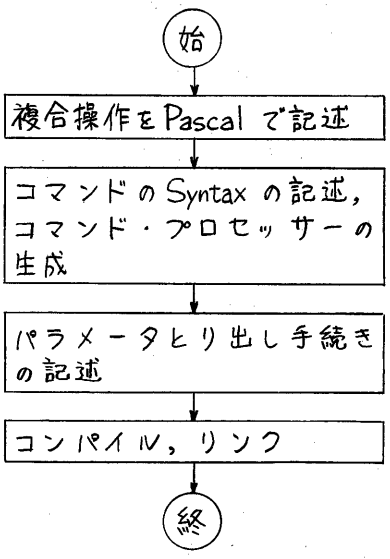

Fig.4 複合操作作成の手順

になる。

```
PROCEDURE PRISM(N : INTEGER; R,Z : REAL; BASE,RET : INTEGER);
    VAR T,X,Y : REAL;
        I,F   : INTEGER;
  BEGIN
    F := NORMALEDGE;
    T := 2 * PI / N;
    FOR I := 0 TO N - 1 DO BEGIN
      X := R * COS(T*I);
      Y := R * SIN(T*I);
      PUTMC(BASE+I,  X,Y,0,F);
      PUTMC(BASE+N+I,X,Y,Z,F)
    END;
    MMBFLV(BASE,RET,0,0,BASE,BASE);
    MMEV(RET,BASE,BASE+1,S,BASE,CCW,BASE,BASE+1);
    FOR I := 1 TO N - 2 DO
      MMEV(RET,BASE+I,BASE+1+I,S,BASE-1+I,CCW,BASE+I,BASE+1+I);
    MMEF(RET,BASE-1+N,BASE-2+N,CCW,BASE,BASE,CW,LEFT,BASE-1+N,0,0);
    FOR I := 0 TO N - 1 DO
      MMEV(RET,BASE+I,BASE+N+I,S,BASE+I,CW,BASE+I,BASE+N+I);
    FOR I := 0 TO N - 2 DO
      MMEF(RET,BASE+N+I,BASE+I,CW,BASE+N+I+1,BASE+I+1,CCW,LEFT,0,0,0);
    MMEF(RET,BASE+N+N-1,BASE+N-1,CW,BASE+N,BASE,CCW,LEFT,0,0,0)
  END;
```

Fig.5　複合操作のPascalによる記述例

② コマンド・シンタックスの記述

本システムではコマンド・プロセッサー・ジェネレータを用いて，メタ言語で記述したシンタックスからコマンド・プロセッサーを生成している。したがって，ユーザはコマンドのシンタックスを決め，メタ言語でそれを記述しなければならない。

先ほどの正多角柱の例に沿って考える。コマンドのシンタックスを，BNF記法にならった形で，

<コマンド> ::= PRISM <N><R><Z> BASE(<BASE>) RETURN(M(<RET>))

とする。<…>は非終端記号とし，<N>，<BASE>，<RET>は符号なし整数，<R>，<Z>は実数とする。各非終端記号の意味は①で説明したとおりである。

このシンタックスをメタ言語で記述すると，Fig.6の様になる。'$'に続く語はメタ言語の予約語である。主な意味を順に説明すると，まず各

```
$COMMAND PRISM   $IGNORE '('')'
                 $LEVEL1;
                   $A;
                   $R;
                   $R;
                   BASE($A);
                   RETURN(M($A));.;
                 $END;
```

Fig.6　コマンド・シンタックスの記述例

コマンドの定義文は$COMMANDで始まり，$ENDで終了する。$IGNOREは引用符で囲まれた記号が単なるセパレータで，パラメータとしてとり出されないことを示す。$Aは符号なし整数を，$Rは実数を表す。
このシンタックスをコマンド・プロセッサー・ジェネレータへ入力して，コマンド・プロセッサーのソース・プログラムを生成しておく。

③ パラメータとり出し手続きの記述
シンタックスに対応して，パラメータをとり出し，①で作った手続きを呼ぶ手続きを記述する。正多角柱の場合は，Fig.7の様になる。

```
PROCEDURE COMPRISM;
    VAR N,BASE,RET : INTEGER;
        R,Z        : REAL;
  BEGIN
    QOPRD(INT,RRR,IDENT,ISTR,IATTR,INO,IFLAG,ITOKEN);
    N := INT(.1.);
    R := RRR(.2.);
    Z := RRR(.3.);
    BASE := INT(.5.);
    RET  := INT(.8.);
    IF N <  3   THEN N := 3;
    IF R <= 0.0 THEN R := 1.0;
    IF Z <= 0.0 THEN H := 1.0;
    PRISM(N,R,Z,BASE,RET);
    DISP; VIEWINF1
  END;
```

Fig.7 パラメータとり出し手続きの記述例

④ コンパイル・リンク
①～③で書きかえたり，生成したプログラムをコンパイル・リンクする。この操作はTSSにおけるOS管理下のコマンド・プロシージャとして，カタログされているので，ユーザの細かな介入は必要ない。

4．実験結果と考察

4．1 ユーザ定義による利点

ユーザは実際の仕事をするときには，オイラー・オペレーションのような基本的な低レベルの操作はほとんど用いず，もっと高レベルの操作を多用すると考えられる。したがってこのように複合操作の定義をユーザに解放することによって，ユーザはかなり自分の目的に合った型のシステムを構成できる。また多くの複合操作を用いることによって，複雑な形状も少ない操作で作ることが可能となる。
本システムには現在，複合操作としてPRISM（正多角柱を作る），CONE（正多角錐を作る），LAMINA（厚さ0の膜を作る），SWEEP（平行掃引を行なう）等と，複合操作を助ける操作として，座標計算を行なうCIRCLE（円周），ARCC（円弧）等がある。これらを用いると，Fig.8のような形状は，オイラー・オペレーションのみを用いた場合の約1／5以下の操作で作れる。
現在は上述のように多角形を中心としたシステムであるが，ここで述べた複合

操作作成の手順はくり返し適用可能であり、システムの機能のいっそうの拡大ができる。

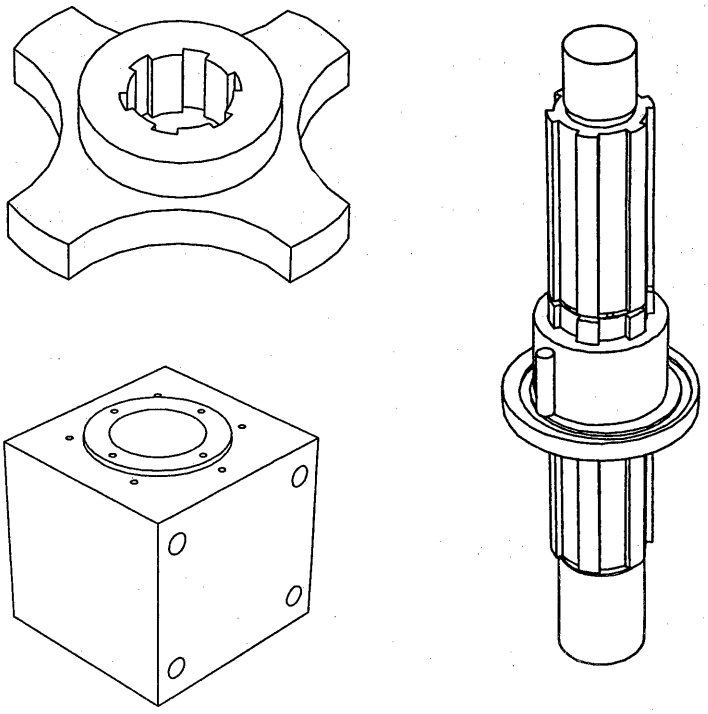

Fig.8　複合操作で作成した形状の例

4.2　ソフトウェア・ツール

一つの独立したソフトウェア・ツールとして，コマンド・プロセッサー・ジェネレータを用いたために，ユーザにコマンドのシンタックスを解放することがかなり容易になった。このコマンド・プロセッサー・ジェネレータの特徴として，

(i)　文法定義が比較的容易である。
(ii)　ユーザ定義のコマンド全体について，キーワードを分析し，あいまいさのない最短の省略形が求まる。
(iii)　コンソール，ディスク等の入出力を管理しているため，エコー・バック等出力の制御や，コマンドのヒストリーをとることもできる。

などが挙げられる。

しかし，機能的に改良を要する点もある。たとえばシンタックスの非終端記号が表現できないため，複数の場所で現れる同一の非終端記号に対応する文法表現をその都度記述しなければならない欠点がある。そこで，この欠点はマクロ・プロセッサーを用いてカバーした。しかし，実際には，この様な機能もコマンド・プロセッサー・ジェネレータに含ませるべきである。

マクロ・プロセッサーはソフトウェア・ツールとしては，コマンド・プロセッ

サー・ジェネレータに比べると汎用性の高いもので，本システム作成の過程においても頻繁に用いたが，これを本システムの本質的な部分へ積極的に用いることも可能である。

試作システムでは，ユーザが複合操作を作り，システムに登録する部分はユーザがPascalで一つの手続きを記述することになるが，ここにマクロ・プロセッサーを用いて，一つの複合操作に対応する基本操作列をユーザが定義し，複合操作のコマンドが入力されるたびに，マクロ展開を行ない，システムはそれを基本操作の列として解釈するという方法も考えられる。この方法をとると，ユーザはPascalのメイン・プログラムを書きかえないので，これについて，コンパイル等をしなおす必要がない。しかし複合操作で用いることのできる機能は限定される。たとえば，Pascalの機能が使えないため，パラメータの拘束条件等のチェックができない等の欠点がある。

現在，マクロによる定義方法についても考慮しながら，コマンド・プロセッサー・ジェネレータをより便利で高機能なものにするために，仕様等を再検討中である。

5．まとめ

生産ソフトウェアにおいて，中心的位置にある形状記述システムにおいて，ユーザが独自の操作及びコマンドを作成・付加可能なシステムを試作した。

(1) コマンド・メモリーという概念を導入することにより，オイラー・オペレーション間のパラメータ受け渡しが容易になり，複合操作の実現が容易になった。

(2) 複合操作のユーザへの解放によって，ユーザは少ない労力によってより自分に合ったシステムを構成できるようになった。また，このような複合操作をうまく用いることによって，基本操作（オイラー・オペレーション）のみの場合に比べて，かなり少ない操作（本文中の例では約1／5）で形状を作成できた。

(3) 本システムの作成にあたって，コマンド・プロセッサー・ジェネレータを用いたことにより，ユーザへのコマンド・シンタックスの解放が容易に可能になった。今後，このコマンド・プロセッサー・ジェネレータを改良するだけでも本システムの機能はかなり向上すると考えられる。

参考文献

1) I. C. Braid et al.: Stepwise Construction of Polyhedra in Geometric Modelling, Mathematical Method in Computer Graphics and Design, Academic Press, (1980) 123.
2) M. Mantyla & R. Sulonen: GWB: A Solid Modeler with Euler Operators, IEEE CG & A, 2-2, 17/31, (1982).
3) 小島俊雄 他：コマンドプロセッサジェネレータとそのCADシステムへの応用，情報処理学会 グラフィクスとCAD研究会，11－3，(1983).