# 対話形グラフィック・シミュレーション・システム*

金田悠紀夫** 島田俊夫**

**Abstract**

This papar describes an interactive graphic simulation system called EGSS by using flowchart input.

This system was designed that simulation models are built and tested incrementally.

The object of incremental simulation is to improve the efficiency of building models by removing the distinction between the building and testing phases.

We also discuss about problems of data structures which is important in such graphic application system.

## ま え が き

コンピュータのシステム・シミュレーション分野への応用は，コンピュータ・システムの大形高速化と，GPSS や SIMSCRIPT などのシミュレーション言語の発達普及にともなって発展し，コンピュータ応用の極めて重要な分野の一つとなっている．

しかし現在広く用いられているシミュレーション言語にはオンラインで対話しながらシミュレーションを進めるという機能が乏しく，プログラム作製，デバッグ，テスト，変更を能率よく行えない欠点がある．

コンピュータと人間の対話の重要性が認識されてくるにつれて，コンピュータの歴史に２つの技術——TSS，コンピュータ・グラフィックス——が登場してきた．これらの技術をシミュレーションに取入れようとした試みに CTSS で行われ Multics で拡張された OPS-3，OPS-4[2] と呼ばれるシミュレーション言語がある．この言語はシステムと対話しながらシミュレーションを行える点画期的であるが，モデル作成に図形情報を用いることは含まれていない．

グラフィック・システムは複雑なシステムの構造を図形で表示でき，それを理解するのに大変有効な手段である．特にシミュレーションの動きをモニタしながら，ステップ・バイ・ステップにモデルを作成し，変更し，テストしていくのに大変適している．

そこでわれわれは，端末から対話的に図形情報を用いてモデルを作成し，シミュレーションを行えるシミュレータを開発した．

### 1. 対話形モデル作成の特徴

複雑な問題を研究する場合それらをいくつかの部分に分解し，互いの関係を階層状に構成して分析すると比較的容易にモデルを記述，解析できる場合が多い．

この場合モデルを組立てる方式には，２つあって，１つは細部を作り，それらを結合して大きくまとめ上げる方式と全体の大きな構造を先に作り，その後で各部分を仕上げていく方式とがある．

これらの方式にはそれぞれ利点，欠点があり，実際のモデリングには両方の手法をうまく組合わせることによって行なっている．本シミュレータにおいてもこれら両方のモデリング手法が生かせることを特に配慮している．

### 2. システムの概説

前節で述べた点を考慮して，われわれはグラフィック装置を用いたオンラインシミュレータの実験システムを開発したが，以下その概要を述べる．

#### 2.1 シミュレーション言語

シミュレーション言語としてすでに開発されているものは数多くあるが，

1. 広く一般に普及している．
2. モデルをブロック図形で組立てて入力すること

---

* Interactive Graphic Simulation System, by Yukio Kaneda and Toshio Shimada (Computer System Section, Computer Division, Elctrotechnical Laboratory)

** 電子技術総合研究所電子計算機部計算機方式研究室

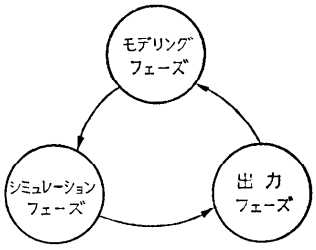


図 1 シミュレータの3つのフェーズ
Fig. 1 Three phases of the simulator

が可能.

という2点を考慮して，言語としては GPSS を基本とすることにし，それに複雑なシステムのモデルを限られたディスプレイ画面上に表示可能にするため新しく Box という特別なブロックを加えて言語を拡張し多数のブロックからなるモデルも容易に入力できるようにした.

### 2.2 シミュレータの構造

本シミュレータは大きく分けると3つのフェーズ——モデリング・フェーズ，シミュレーション・フェーズ，出力フェーズ——から成り，ユーザは制御プログラムに指示を与えることによってこれらのフェーズを切換えてシミュレーションを行う.

モデリング・フェーズ——モデルをブロック図で組み立てる.

シミュレーション・フェーズ——モデリング・フェーズで作られた結果を用いてシミュレーションを行う.

出力フェーズ——シミュレーション・フェーズで得られた結果をディスプレイ上に表示する（図1）.

### 2.3 Box の概念

一般に GPSS で組立てられたシミュレーション・モデルは数百個のブロックを含んでいる．ディスプレイの1画面に描けるブロック数はせいぜい 20 程度であまり多く描けない.

したがって何枚かの画面に分けて表示する必要がある．その方式としては全体を1枚の大きな画面とみなし，それを何枚かのサブ画面に分割し，番号をつけて管理しておく方式が考えられる.

しかしこの方式だとモデルの表示が一面的になりすぎ，画面間の論理的なつながりという面が薄くなる欠点があり，人間のモデル作成の手法と適合しにくい.

そこで前述のモデルの持つ階層性に注目し，これを積極的に生かし，モデルを Box と呼ぶエレメント群

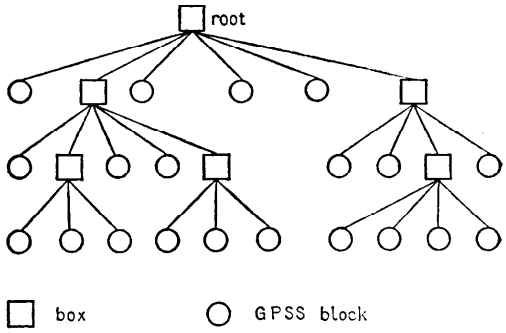


図 2 モデルのトリー構造
Fig. 2 Tree structure of a model

に分割し，モデルを Box の集合であらわし，それら Box の中身を記述していくことによりモデル全体を記述するという考え方を採用した（図2）.

Box はディスプレイ上の一つの画面を表わし，中身はブロック群の集合となっており，各 Box は名前が付けられている．またこれらの Box は Fig. 2 のようにトリー構造に構成されており，Box 内に Box が含まれていてもよいが，同一名を持つ Box の重複使用は許されない.

図2からも明らかなように，Box は1つの画面に当り，トリー構成の要素としてはブランチ・ノードとなっている．また GPSS・ブロックはターミナル・ノードとなっている.

各 Box は論理的にはサブ・トリーの全ブロックを含んでおり，したがって root Box は全モデルを示すことになる.

### 2.4 モデルの保存

本システムは長期にわたってモデルを保存するため EDSP と呼ばれるファイル・システムを含んだデータ構造パッケージを利用しており，ターミナルから作製したモデルを保存しておき，任意の時点でそれにアクセスしてモデルの作製やシミュレーションを再開できるようになっている.

## 3. システムの構成

### 3.1 制御方式

本システムは独立な5つのモジュール——EGSS, GSS, SLU, GPSS, OUTPUT——から構成されており，EGSS が他の4つのモジュールの制御を行なっており，モジュールの実行指令，モジュール間の情報の受け渡しの制御をしている.

各モジュールの機能をジョブの流れに従って説明し

ていく.

EGSS は HITAC-8400 オペレーティングシステムの持つキーイン・アウト・シミュレーションの機能を用いて，ユーザの指示に従って，オペレータの介入なしに自動的に GSS 等のモジュールのロードと実行を制御している.（この場合，EGSS と他のモジュールはマルチプログラミングモードで走る.）

まず EGSS は GSS をロードし実行させる.

GSS はディスプレイ上にシミュレーション・モデルを作製する際のユーザとグラフィック装置間の情報交換をすべて管理している.

モデルの作製が終了したとき，あるいは作製途上でテストが必要なときに，ユーザがシミュレーションの実行を要求すると，GSS は終了しコントロールは EGSS に返される.

EGSS は SLU をロードして実行し，つづいて GPSS モジュールをロードしてシミュレーションの実行をする*.

シミュレーションが終了すると EGSS はつづいて OUTPUT モジュールをロードし，GPSS の出力結果を編集してディスプレイ上に表示する.

ユーザはその結果を見て，さらにモデルの変更をしたいときは，再び GSS をロードし，ディスプレイ上に表示された先程のモデルを適当に変更して，再び実行を指示するという手順を繰り返せばよい.

結果が満足なものであれば，モデルをファイルに残しておくかどうかを指示してシステムを終了させる.

なお，われわれの用いたハードウェア構成を図 3 に示しておく.

### 3.2 入力方式

ユーザはモデリング・フェーズでモデルを組立て，システムに新しく入力したり，すでに入力されたモデルに対する変更を行うことができる.

したがって，ユーザはこのフェーズで新しいプログラムの追加や変数の入力，ブロック間の結合が行えるほか，入力ずみのブロックやそれらの結合関係の消去，変更が行えるようになっている.

これらの入力操作はすべて，グラフィック装置の，ライトペン，英数字キーボード，ファンクション・キーボードを用いて行われている.

#### 3.2.1 ファイル定義とモデルの構造

GSS はロードされると，ユーザは自分のファイル名とそのファイルが，今新たに作成されたものか，あるいはすでに作成してファイル・ライブラリーに登録されているものかを示すコードを入力する.

ファイルが新しく作成されたものであると，GSS は図 4 で示す BASE 画面を表示する.

また，もしファイルがすでに作成されたものであると，そのファイルの BASE 画面が呼出され表示される.

BASE 画面は，root となる画面で，この画面を他の画面に従属する関係で連結することはできない.

#### 3.2.2 モデルの作成

BASE 画面でライトペンで CREATE をピックすると create モードとなり，BASE 画面の右側に GPSS のブロックを表示する．その中の一つをライトペンでピックし，任意の点に移動することによって該当するブロック図を指定した場所に表示させる.

次にそのブロックの変数を英数字キーボードからタイプインする．これら作られたブロックを目的に応じ

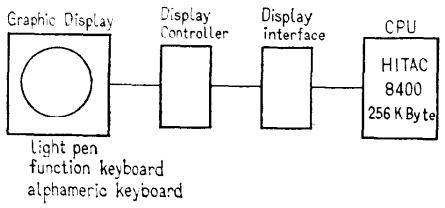


図 3 システムのハードウェア構成

Fig. 3 Hardware Structure of our system

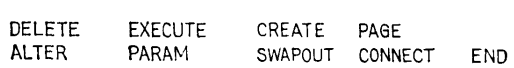


図 4 Base 画面

Fig. 4 Base Frame

* SLU, GPSS モジュールはいずれも日立製作所が提供しているプログラムを直接利用しており，SLU は GSS の出力を GPSS の入力に編集している.

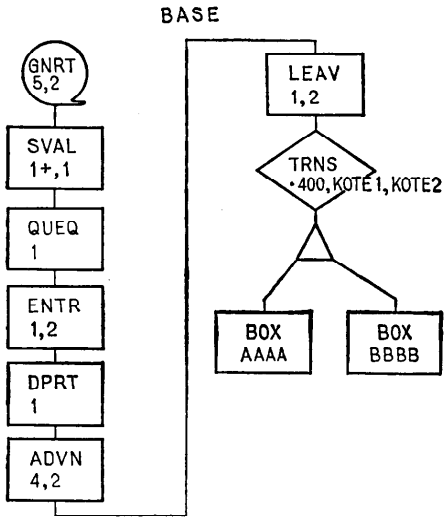


図 5 Base 画面のモデル作製例
Fig. 5 Example Model in Base Frame

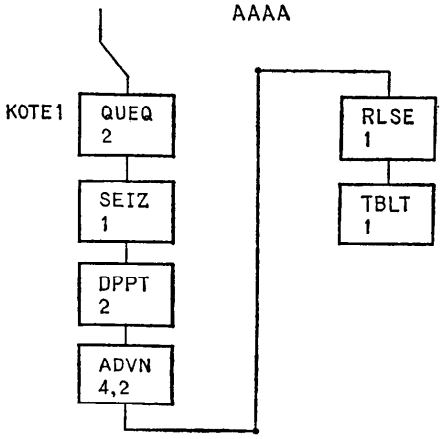


図 6 AAAA 画面のモデル作製例
Fig. 6 Example Model in AAAA

て，ライトペンの指示により CONNECT（連結）していくことにより，モデルを作っていく（図5）．

画面が一杯になる前に，Box ブロックを作って，変数として Box 名を付けて，新しい画面を指定しなければならない（図5～図8）．

Box 画面でのモデルの組立ては BASE 画面の場合と同様な方式で行われる．またブロックやその結合関係の消去 (DELETE) や変数の変更 (ALTER) はモデル作成中，任意の時点で行える．

PAGE はすでにモデルを作成した画面（BASE 画面，Box 画面）を随時参照するために用いる．

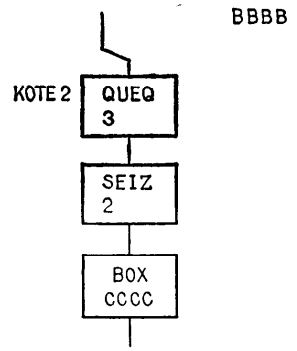


図 7 BBBB 画面のモデル作製例
Fig. 7 Example Model in BBBB

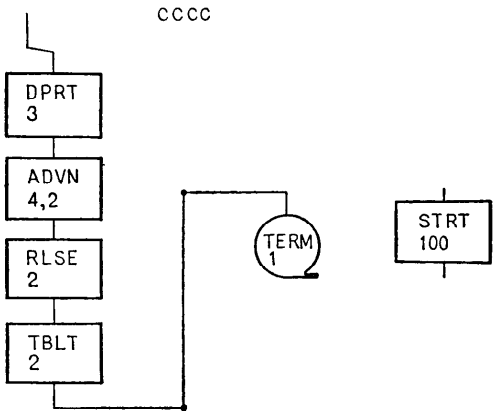


図 8 CCCC 画面のモデル作製例
Fig. 8 Example Model in CCCC

ユーザは，モデルを SWAPOUT しておけば，好きな時に，モデルをファイル・ライブラリーから引出して仕事を再開することができる．

### 3.2.3 システム変数の入力

GPSS には FUNCTION とか VARIABLE とか，TABLE などのシステム変数がある．

これらの変数は，前述の方法では入力することができないので，別にシステム変数入力用画面を用いて入力している．この場合もブロック図の入力方式と同様変数の作成，変更，削除をライト・ペン，ファンクション・キー，英数字キーボードから容易に行えるようになっている．

### 3.3 シミュレーション実行

ユーザがモデリング・フェーズでシミュレーション実行の指令を出すと GSS はユーザの作成したモデル（木構造形成で EDSP 上に保存されている）をたど

```
2          TABLE       M 1,4,1,15
1          TABLE       M 1,4,1,15
           GENERAT     5,2
           SAVEALUE    1+,1
           QUEUE       1
           ENTER       1,2
           DEPART      1
           ADVANCE     4,2
           LEAVE       1,2
           TRANSFER    .400,KOTE 1,KOTE 2
KOTE 1     QUEFE       2
           SEIZE       1
           DEPART      2
           ADVANCE     4,2
           RELEASE     1
           TABULATE    1
           TERMINATE   1
KOTE 2     QUEUE       3
           SEIZE       2
           DEPART      3
           ADVANCE     4,2
           RELEASE     2
           TABULATE    2
           TERMINATE   1
           START       100
```

図 9　生成されたプログラム

Fig. 9　Generated Program

り，その結合関係を利用してカード形式の GPSS プログラムを発生して磁気テープに出力して GSS の実行を終る．SLU の処理により，GSS の出力テープの編集を行い，GPSS の入力形式に編集した後，GPSS の実行に移る．

GPSS はこの出力テープを続んで，シミュレーションの実行を行い，結果を別の磁気テープに出力した後終了する．

GPSS が終了すると OUTPUT プログラムが実行され出力フェーズに移る．

### 3.4　出力方式

対話的にシミュレーションをグラフィック・ターミナルから行う場合，シミュレーションの結果を出力する方式は入力方式とともに極めて重要な役割を果す．

持つべき機能としては，必要なデータを検索できかつ整理した形式で表示できること，重要なデータはファイルとして保存し，後で参照できることなどがある．

OUTPUT プログラムは GPSS の出力から，必要な情報を検索しディスプレイ上に編集して表示する．

したがって，ユーザはリストの内容，各種統計データの内容などをグラフィック・ターミナルからの指示を出すことによって必要な情報を得ることができる．

表示例を図 10，図 11 に示す．

また必要なデータは，EDSP 上にファイルとして保存し必要に応じて呼び出し，様々な結果を比較し，必要なときには，ハード・コピーとしてユーザの指示により，入力リスト，結果などをラインプリンタ上に出力する機能も有している．

### 3.5　部分実行

モデルをサブモジュールに分割して作製，テストを行い，全体のモデルを完成させていくため，本システムでは Box 単位でシミュレーションの実行ができるようになっている．したがって，各サブモジュールを

```
                    ****  GPSS      OUTPUT    ****

BLOCK
NUMBER        *LOC      OP              A, B, C, D, E, F, G      COMMENTS      CARD #
                        JOB                                                    00001
                        REALLOCATE      200 K                                  00002
                  *                                                            00003
                  *                                                            00004
                  *                                                            00005
                  *                                                            00006
                  *                                                            00007
                   1      STORAGE       4                                      00008
                   2      TABLE         M 1,4,1,15                             00009
                   1      TABLE         M 1,4,1,15                             00010
00001                     GENERATE      5,2                                    00011
00002                     SAVEVALUE     1+,1                                   00012
00003                     QUEUE         1                                      00013
00004                     ENTER         1,2                                    00014
00005                     DEPART        1                                      00015
00006                     ADVANCE       4,2                                    00016
00007                     LEAVE         1,2                                    00017
00008                     TRANSFER      ,400,KOTE 1,KOTE 2                     00018

INPUTLIST                                                    TOP    NEW    STOP
```

図 10　出　力　の　例

Fig. 10　The example of output

**** GPSS OUTPUT ****

| QUEUE NUMBER | MAXIMUM CONTENTS | AVERAGE CONTENTS | TOTAL ENTRIES | ZERO ENTRIES | PERCENT ZEROS |
|---|---|---|---|---|---|
| 1 | 1 | 0.000 | 101 | 101 | 100.00 |
| 2 | 1 | 0.050 | 60 | 50 | 83.33 |
| 3 | 1 | 0.029 | 40 | 32 | 80.00 |

| QUEUE NUMBER | AVERAGE TIME/TRANS | AVERAGE TIME/TRANS | TABLE NUMBER | CURRENT CONTENTS |
|---|---|---|---|---|
| 1 | 0.000 | 0.000 | 0 | 0 |
| 2 | 0.433 | 2.600 | 0 | 0 |
| 3 | 0.375 | 1.875 | 0 | 0 |

TOP NEW STOP

QUEUE

図 11 出 力 の 例

Fig. 11 The example of output

Box に対応させておくことにより，サブモジュール毎のテストが行える．任意の Box を指定してシミュレーション実行の指令を出すと，その Box が root Box* として取扱われシミュレーションが実行され結果が出力される．（システム変数は全モデル共通である．）

このようなテスト，修正のサイクルを繰返すことにより，モデル全体の作製とチューニングの円滑化をはかることができる．

## 4. データ構造

### 4.1 図形表示用データと問題記述用データ

グラフィックシステムを用いて対話形システムを構成する場合，図形表示用データと問題表示用データを，どのようなデータ構造で表現するかという問題がある．

両方のデータを同一のデータ構造で表現した場合は，アプリケーション・プログラムの作成において，図形表示の部分と問題処理の部分を一体化することができるが，性質の異る両データを取扱うためデータブロックとして可変長のものが要求されるなど，データ構造が複雑化し，多くの場合，問題処理の能率も低下すると考えられる．

これに対して，図形表示用データと問題記述用データを別々のデータ構造で表現した場合，個々のデータ構造が簡単になる．アプリケーションに応じて各々が適当なデータ構造を利用できる．図形表示と問題処理を別々の計算機で実行させることが容易なので，TSS に向くなどの利点がある．

しかし，この方法は，問題を処理している最中は対話の必要のないシステムには有効であるが，たとえば，高度のオンライン・インタラクティブ・シミュレーションでは，シミュレーションの実行中にモデルを変更し実行を続けることがあるが，このように問題処理の最中に対話が必要な場合には図形表示用データと問題記述用データの関係がより繁密化するので，同一のデータ構造で表現した方が良いと考えられる．

EGSS のデータは，図形表示用は，図形の挿入，削除，修正，サブピクチュアの定義などを伴い，かなり複雑な階層をなす構造が必要であり，問題記述用はブロックの属性，サブ画面間およびブロック間の接続関係を表わし，図形表示用データと同様，挿入，削除，修正などが必要であるが，構造は比較的簡単である．また問題記述用のデータが創られたり，変更されたりするのは，ユーザがディスプレイ上で図形操作を行なった時のみで，問題処理の最中，すなわち，シミュレーションの実行の際には，問題記述用のデータのみを使用すればよい．

したがって EGSS では，これらのデータを，二つのデータ構造を用いて表現するのが適当であり，図形表示用には GSP (Graphic Subroutine Package[1)5)]), 問題記述用には EDSP (ETL's Data Structure Package[4)]) を用いている．

GSP は FORTRAN のサブルーチンとして使用するグラフィックシステム用言語である．

EDSP は，ASP あるいは LEAP などと同等の機能を持つ言語で，連想３つ組によって情報を表現し，この３つ組に対する演算によって，検索，集合演算な

---

* その Box に含まれる GPSS ブロックだけで独立してシミュレーションの実行可能となるように事前に使用者が Box の内容を修正する必要がある．

どを行う.

### 4.2 図形再表示に関する問題

すでに述べたように，EGSS では，シミュレーションモデルを何枚かのサブ画面に分割し作製する方法をとっている．したがって，あるサブ画面をつくっているときに，すでにつくった他のサブ画面を参照したいという要求がしばしばおこる．対話形のシステムではこのような要求を迅速に処理せねばならない*1.

これは図形表示用データをどのような形で保存するかという問題につながる．これらのデータ量は通常膨大なものであり，保存の形式として二つの方向が考えられる．一つは，できるだけデータを簡単化，抽象化し，図形を再表示するときには，適当な変換ルーチンを通して図形を再生する．この方法は，データ量は少なくなるが，変換ルーチンが複雑になり，画面の再表示に時間がかかる*2．も一つの方法は，できるだけデータを生のまま取り扱う方法である．この方法は，データ量は膨大になるが，画面を再表示するのに簡単な変換ルーチンでよく，再表示までに時間がかからない*3.

どちらの方法をとるかは，各システムの目的や，要求される処理の効率，hardware, software の制約などがあるので一般的に論じることは難しい.

EGSS は，図形データをできるだけ生のまま取り扱って処理の効率をあげている*3.

## 5. 結　論

本研究を進めるにあたって得られた結論は大別すると二通りある．一つはオンライン・グラフィック・シミュレーションに対する評価であり，もう一つは，グラフィックシステムを用いて対話形のシステムを構成する際のデータ構造の問題である.

実用レベルの大規模システムのシミュレーションを行う場合，作製途中のモデルの保存，全体の構造の見通しの悪さ，デバッグの能率などがネックになりがちであるが，本システムでは次の機能

1. ファイルとしてモデルの構造やシミュレーション結果を保存しておき必要なとき利用できる.
2. Box ブロックの導入により，incremental にモデルを作り上げることが可能である.
3. モデルを図的情報で，モジュール化，階層化して記述できるのでモデルの構造を容易に理解できる.
4. Box 単位のシミュレーション機能を用いて，システムのデバッグ，最適化が能率よく行える.
5. 出力結果を保存し，これらのデータを別のプログラムで処理することにより，結果をグラフ化したり，モデルの最適化処理を行うことが可能である.

を持つことによりこれらの問題を解決している.

また，データ構造の問題については，

1. 本システムのように，取り扱うデータが図形表示用と問題処理用に明確に区別でき，しかも問題処理に際しては対話の必要がなく，問題処理用データのみを使用する場合，すべてのデータを一つのデータ構造で表現せずに，各々異なったデータ構造を用いれば，データ構造が簡単化し，アプリケーションに応じて最適なデータ構造を利用することによって問題処理の効率を上げることができる.
2. 図形表示用データの保存の形は，対話形であることを考慮するなかで，各システムの目的や制約に応じて適当な形式を考える必要がある.

などである.

なお実用規模のモデルとして 100～200 ブロックからなるモデルを作製した経験では，オンラインファイルの役割が極めて重要で，図形情報は多量となるのでファイルは大容量でなければならないこと，新しいブロックを作ったり，既存のブロックを変更したり消去したりするたびにファイルに対してアクセスするのでファイルアクセスが高速に行えること，また長時間大量の情報を格納するため信頼性が十分高いことなどが必須であることが判明した.

本グラフィックシミュレータには，シミュレーション中に，オンラインにシステムと対話して，シミュレーションの動きをモニタし，モデルの構造を変更することはできないが，この性能が加われば，より一層有力なシミュレーション手法になるものと思われる.

本システムで目指したシミュレーションの方式は TSS の目標とする方向と極めてよく一致しており，端末として，ディスプレイターミナルを持ったシステムの応用プログラムとしては極めて有効なものと考えられる.

---

*1 実用性からみて 10 秒以内であることが望ましい.
*2 簡略化の程度によるが，われわれがテストした一つの例では再表示に 30 秒程度かかる.
*3 EGSS では 2～3 秒である.

## 謝 辞

本システム作成に協力して頂いた，システム開発株式会社の宮島実氏，矢野修氏に深謝するとともに，本研究の推進に有益な御指導を頂いた黒川一夫電子計算機部長，西野博二ソフトウェア部長をはじめ情報システム研究室の諸氏に感謝します．

## 参考文献

1) Baskin, H. S. and Morse, S. P.: "A Multilevel Modeling Structure for Interactive Graphic Design," IBM System J. Nos 3 & 4, 1968.
2) Jones, M. N.: "Incremental Simulation on a Time-Shared Computer," MIT Tech. Report, 1967.
3) プログラム・マニュアル GPSS 日立製作所
4) 古川康一，"EDSP について"，昭和45年度情報処理学会大会予稿集.
5) HITAC 8400 グラフィック・サブルーチン・パッケージ説明書，電子技術総合研究所&コンピューターシステム社.
6) 大須賀節夫："コンピュータ・グラフィックス〔1〕〔2〕"，情報処理，Vol. 12, No. 4 & 7, 1971.
7) M. F. Robbins and J. D. Beyer: "An Interactive Computer System Using Graphical Flowchart Input," CACM, Vol. B, No. 2, Feb. 1970.
8) R. L. Beckermeyer,: "Interactive Graphic Consoles-Environment and Software," Proc. FJCC, 1970.

## 6. Appendix——例

EGSS を使って行なったシミュレーションの簡単な例を示す．モデルの作製はモデリング・フェーズでライトペン，ファンクション・キー，タイプライタを使って，まず基本となる BASE 画面上に，GPSS フローチャートブロックを用いて行う（図5）．

BASE 画面が一杯になったときは，Box ブロック（この例では AAAA と BBBB）を作り，以後はこの Box ブロックに対応するサブ画面にモデルを作る（図6, 7）．

さらにサブ画面が一杯になれば，また Box ブロック（この例では CCCC）を使って，CCCC 画面に以後のモデルを作ればよい（図8）．

またサブ画面をモデルの機能に応じて作ることによってモデルの構造をわかりやすくすることができる．これは Box を FORTRAN のサブルーチンと同じように使う方法である．

実行を指示すると図9のプログラムが GPSS に入力され，シミュレーションの結果がディスプレイ上に表示される．その一例を図10～11に示す．

(昭和47年3月15日受付)
(昭和47年4月24日再受付)