Vol. 37 No. 3　　情　報　処　理　　Mar. 1996

解　説

シミュレーション技術の最近の動向

2.2 離散系シミュレーション言語 SIMSCRIPT II.5†

中　西　俊　男††

1. SIMSCRIPT概説

SIMSCRIPT(SIMulation SCRIPTor)は，GPSSと同じく1961年アメリカのRand Corporationで，M. Harry Markowitz, Bennard Hausner, Herbert W. Karrらによって開発された汎用の離散系シミュレーション言語である．初版はFORTRANをホスト言語にしたもので，SIMSCRIPTで書かれたシミュレーションプログラムはいったんFORTRANのプログラムに変換され，その後コンパイルされて実行された．したがってそのプログラムの中にFORTRANで書かれたプログラムの混在が許された．つまりこれは，SIMSCRIPTが，シミュレーション専用というより汎用のプログラミング言語としても使えたということを意味している．

SIMSCRIPTは，その後バージョンアップを重ね，現在米国CACI, Inc.のSIMSCRIPT II.5が最新のバージョンとして利用されている．IBM, CDC, UNIVAC, Perkin-Elmer, Prime, NCRなどのメインフレームで使えるほか，IBM—PCを初めとする各種PCにインプルメントされ，広く用いられている．なお初期のSIMSCRIPTは，FORTRANで書かれていたが，SIMSCRIPT II.5は，メインフレーム，PC用ともにC言語で書かれている．

初期のSIMSCRIPTは，事象中心(event-oriented)のモデル構成概念(world-view)であったが，SIMSCRIPT II，同II.5からプロセス中心(process-oriented)のモデル構成概念に移った．現在では，SIMULAなどとともに，代表的プロセス中心の離散系シミュレータとなっている．

SIMSCRIPT II.5は，汎用プログラミング言語としての機能も兼ね備えている関係で，これを用いてほかの専用目的のシミュレータを作ることも可能である．たとえば，FA(工場の自動化)用シミュレータSIMFACTORY II.5，通信ネットワーク用シミュレータCOMNET II.5, NETWORK II.5などがその例である．またSIMSCRIPT II.5には，アニメーション機能も備わっているので，これらシミュレーションパッケージは，すべてオンライン(インタラクティブないしビジュアル)シミュレーション機能を具備したオブジェクト指向言語となっている．

本稿では，以下SIMSCRIPT II.5のモデル構成概念，プログラム構造，variable，プログラム文，中でも重要なevent(**6.**)，process(**7.**)とその変遷(**8.**)について特に詳しく述べた．さらに**9.**で付加機能であるアニメーション機能SIMANIMATIONについて触れた．

2. SIMSCRIPT II.5[1),3)]のモデル構成概念

SIMSCRIPT II.5は，プロセス中心のモデル構成概念を持つ．

プロセスとは

"A time-ordered sequence of interrelated events separated by passages of time (either predetermined or indefinite), which describes the entire experience of an "entity" as it flows through a system"

「あらかじめ定められた，もしくは不定の時間経過によって分離された相互に関係あるイベントを時間順に並べたもので，エンティティ(システム構成要素)がシステムを流れる全過程を記述する．」

と定義されている．イベントとは，その中に時間

† A Discrete System Simulation Language SIMSCRIPT II.5 by Toshio NAKANISHI (Seikei University).

†† 成蹊大学

経過を持たない単なる事象であるのに対して，プロセスとはその中に時間経過を含む現象(行動ないし活動)を記述するものなのである．SIMSCRIPT II.5では，プロセスもエンティティと考えられており，process entityと称されている．なおプロセス，イベント，エンティティなどは，SIMSCRIPT II.5のプログラムでは，英語表現で出てくるので以後process, event, entityの形で表現することにする．

ある特定のprocessを記述するサブルーチンをprocess routineと呼んでいる．時間経過を含まないprocess routineは，event routineとも考えられるわけで，processはeventを含む概念と見ることができる．

GPSSのプログラムは，process routineに対応するが，そこを流れるtransactionは，それぞれにprocessを構成し，それぞれがprocess entityを構成している．SIMSCRIPT II.5のモデルでは，このprocess entityの実現が，process noticeと呼ばれるレコードを保持している．簡単にいうと，process noticeとは，生起するprocessに対してつけられた名前および関連情報と考えてよい．processのタイミングのコントロールは，process noticeを対象に行われる．

processの進行中，process entityが必要とするシステム要素をresourceと称している．resourceの実態は恒久要素すなわちpermanent entityに近い概念(厳密にいうとやや広い概念)と見られるが，別ないい方をするとresourceとは，permanent entity, setおよび利用競合による時間的遅延の概念を結びつけたものといえる．

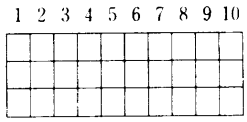
改めてentityについて述べると，entityとは実際の世界の構成要素を表わすデータアイテムで，permanent entity(恒久要素)，temporary entity(一時要素)，event notice, process notice, resourceからなる．また各entityは，attributeを持ち，あるsetを所有したり，あるsetに属することもあり得る．紙面の都合上これらについて詳説はできないが，簡単に説明すると次のとおりである．

(1) permanent entity

permanent entityとは，その数が一定で不動(生滅がない)のシステム構成要素である．その構造は配列(array)で示され，それぞれのentityに対してattributeが定義できる．たとえば，生産機械(MACHINE)が10台あって，それぞれの機械に対して生産効率(PRODUCTION・RATE)，平均故障間隔(MEAN・TIME・TO・FAILURE)，平均修理時間(MEAN・REPAIR・TIME)というattributeが定義される場合，

Every MACHINE has a PRODUCTION. RATE, and a MEAN. TIME. TO FAILURE, and a MEAN. REPAIR. TIME

と記述される．10台ある生産機械のattributeは10×3の配列として



の形でメモリが割り当てられるが，そのための10台分のメモリスペースは

Create every MACHINE(10)

で確保されなければならない．配列の初期値は自動的に0となる．生産機械の数10は，N・MACHINEというシステム変数で表わされる．

(2) temporary entity

permanent entityはシミュレーション実行中，その数の増減がない，存続し続けるシステム構成要素であるのに対して，temporary entityはその数が変動し，生滅が見られるシステム構成要素を表わす．たとえばあらかじめpreamble(後述)でtemporary entity JOBを

Every JOB has a PROCESSING. TIME
and a REQUIRED. MACHINE
and a DUE. DATE

と定義(attributeとしてPROCESSING・TIME, REQUIRED・MACHINE, DUE・DATEを持つ)しておき，シミュレーションプログラムで

Create a JOB
let PROCESSING. TIME(JOB)＝…
let REQUIRED. MACHINE(JOB)＝…

とすることにより，任意の数のJOB(のコピー)を作ることができる．

(3) set

同類のentityで共通特性のあるものを複数集めて，setとして一括処理の対象とすることがある．setはこれを所有するentity(owner entityあるいはowning entity)とsetに所属するentity

(member entity)で構成される．たとえば

Every MACHINE owns a JOB. LIST
Every JOB belongs to a JOB. LIST
Define JOB. LIST as a fifo set

のごとくである．

set の処理命令には次のようなものがある．

FILE　：set に新しい member を入れる．
REMOVE　：set から特定の member を除去する．
FOR EACH
OF SET　：ある特定の属性値を持つ member すべてを捜索するための制御指示命令．

setに関しては，この他もろもろの処理機能があるが，詳しくは参考文献1)～4)を見られたい．

3. SIMSCRIPT II.5のプログラム構造

SIMSCRIPT II.5によるシミュレーションプログラムは4つのセクションに分かれる．

(1) PREAMBLE

preamble は，宣言情報を持つのみで，実行命令を含まない．SIMSCRIPT II.5のすべてのentity は，preamble で宣言されなければならない．global variable(どの routine でも使える汎用の変数．ある routine のみで使える変数は local variable と呼ばれる)も preamble で宣言されなければならない．

(2) MAIN

シミュレーション実行をスタートさせる部分

(3) ROUTINE

main もしくはほかの routine から呼ばれる routine 群で，process(もしくは event)を記述する routine やほかの一般の subroutine を含む．

(4) FUNCTION

算術計算式を参照するとき呼ばれるもので，preamble で宣言されなければならない．

SIMSCRIPT II.5のシミュレーションプログラムは，英語形式の文で書かれる．命令文，宣言文いずれも改行は自由である．またコメント文は，2個のアポストロフィ(')で始まり，record の終りか，次の2個のアポストロフィで終了する．

4. SIMSCRIPT II.5の variable

SIMSCRIPT II.5の variable は，最初が英文字の英数字文字列(ピリオドを含む)から成る．通常の variable のほかに，あらかじめ定義された variable と constant，自動的につくられる system variable がある．その一部を**表-1**に示す．

5. SIMSCRIPT II.5のプログラム文

(1) 計　算　式

FORTRAN とほぼ同じであるが，計算式の前に LET 文を置く必要がある．たとえば

LET A=B+D/(C−E＊F)＊＊2

のごとくである．

計算の優先順位を示すためのかっこが許される．計算式の長さに制限はなく，かっこも幾重に使用してもよい．

(2) その他の命令文

その他の命令文としては，PRINT 文，READ 文，ADD 文，SUBTRACT 文，IF-Then-Else 文，For-do-loop 文，Until-do-loop 文などがある．

6. Event

event とは，システムの状態を変化させる瞬間的事象で，それ自体に時間消費を含んでいない．temporary entity と同様，create(実態もしくはそのためのメモリの確保)，destroy(同じく解放)があるので，一種の temporary entity として扱われ，event set という特別な system set に属する．event は次の3項目で記述される．

表-1 system variable

カテゴリー	形　式	例	意　　味
Constants	name. C	PI. C	π
		INF. C	∞
Variables	name. V	TIME. V	シミュレーションクロック
		HOURS. V	1日あたりの時間数
Functions	name. F	SQRT. F	平方根
		UNIFORM. F	一様乱数の発生
Generated Attributes	letter. name	U. PUMP	resource のユニット数を示す属性
		N. Q. PUMP	resource PUMP の待行列長

(1) event notice: event の特別のコピーもしくはその例について適切な情報を持つデータ群から成る．あらかじめ定義された attribute として TIME. A：その event が次に生起する時刻，EUNIT. A：外生事象に関する特別情報，set attribute：P. EV. S, S. EV. S, M. EV. S を持っている(図-1)．

(2) event routine: event が環境にどう反応し，これをどう変えるかなど，event のロジックを記述するルーチンである．CAUSE, SCHEDULE, ACTIVATE などの命令で起動が計画され，シミュレーションクロックが計画時刻になると起動される．計画された event は，CANCEL で削除される．

(3) event と同じ名前の global variable: event の特定のものを指定するアイデンティファイアとして使われる．

event によりモデルを記述するときの常形を例1に示す．あらかじめ preamble において event 名を定義し，属性を持つものについては，その属性とタイプを定義しておく．その後 event routine で，その event で記述されるモデルの記述を行う．

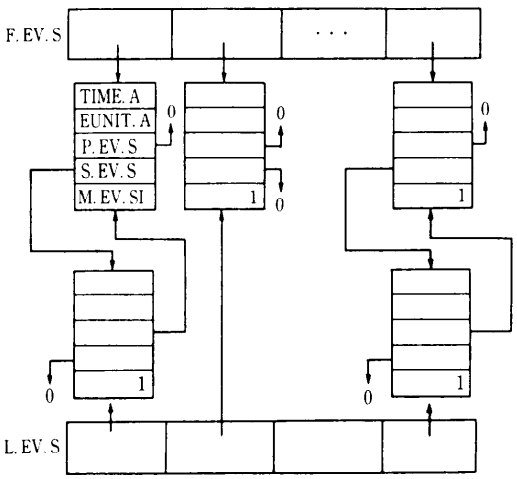


図-1 event set の構造

(例1)

```
event notice include…イベント名の定義
  AIRPLANE. ARRIVAL, LEAVE. GATE
  every BEGIN. TAKEOFF
  has a BT. AIRPLANE…イベント属性の定義
define BT. AIRPLANE as an integer variable
schedule a BEGIN. TAKEOFF now…イベントの起動
event BEGIN. TAKEOFF……イベントルーチン
    .
    .
end
```

(preamble 部に記述)

(例2)

```
event notice の event set からの削除
For each END. TAKEOFF
with ET. AIRCRAFT(END. TAKEOFF) =
MY. AIRCRAFT
find the first case
if found
  cancel this END. TAKEOFF
always
```

図-1 は event set の構造を示したものである．ここで示される用語は次の意味を持つ．

TIME. A：その event が次に生起する時刻
EUNIT. A：外生事象に対する特別情報
P. EV. S：先行 event へのポインタ
S. EV. S：後続 event へのポインタ
M. EV. S：event set への帰属を示す．
F. EV. S：最初の event へのポインタ
L. EV. S：最後の event へのポインタ

また図中の矢印は，ポインタ連結を示す．

7. Process

process は event と違って，その中に時間消費部分を持っている．process はある意味で，一連の event をとりまとめたものといえる．たとえば，次のような例を考えることができる．

```
Processes include AIRPLANE. GENERATOR,
and
  every AIRPLANE has an AC. BEGIN.
WAITING. TIME
define AC. BEGIN. WAITING. TIME as an
integer variable
activate an AIRPLANE now
```

(preamble 部に記述)

```
    .
    .
Process AIRPLANE
  let AC. BEGIN. WAITING. TIME(AIRPLANE)=time.
v
  request 1 RUNWAY
  work uniform. f(min, max, 1)minutes
end"process AIRPLANE
```

このシミュレーションプログラムは，processとして，AIRPLANE. GENERATORとAIRPLANEを持っており，各AIRPLANEは整数のglobal variable AC. BEGIN. WAITING. TIMEを持っている．このプログラムでは省略されているが，process AIRPLANE. GENERATORの中で，process AIRPLANEが現時点(now)でactivate(後述)される．process AIRPLANEは，その属性AC.BEGIN.WAITING. TIMEにTIME.V(現在シミュレーション時刻)をセットし，RUNWAYを1本要求する．そこで所定の一様分布で与えられる時間を消費して，processを終える．

processで使われる命令には次のようなものがある．

activate：あるprocessに起動をかける命令である．scheduleやcauseとほぼ同じであるが，activateにはcreate(メモリの確保)が含まれている．

(例) **activate a CAR in 10 minutes**

(1) 新しいプロセスCARを記述するprocess noticeをcreateする．

(2) variable CAR(自動的に作られる)のこのnoticeに，関係の値をセットする．

(3) この起動がなされる時刻(現在時刻+10分)を計算し，この値を当該processのattributeにセットする．

(4) process noticeをpending list (生起をスケジュールされたprocessのlist) におく．

この場合，つねにCARというプロセスに過ぎないが，processに名前をつけ，

activate a CAR called X in 5 minutes

activate a CAR called Y in 10 minutes

として，processの異なるコピーを複数個生起させることができる．

request：当該processが使うresourceを要求する命令である．

relinquish：使っていたresourceを開放する命令である．

これら命令は，resourceと深い関係があるので，resourceに関してさらに詳しく述べておく．

resourceがcreateされると，次のvariableが自動的に定義される．

(1) N. resource：resourceの最大個数

(2) U. resource(i)：i 番目 ($1 \leqq i \leqq$ N. resource)のresourceの利用可能な容量を示すattribute

(3) N. X. resource(i)：resource(i)を使っているrequestの数を示すattribute

(4) N. Q. resource(i)：現在resource(i)をrequestして待たされている数を示すattribute

(例) **Resource include RUNWAY—preamble部に記述**

create every RUNWAY(1)…1個RUNWAYを作る．すなわちN.RUNWAY(1)=1

let U. RUNWAY(1)=2…RUNWAYの大きさを2とする．

request 2 RUNWAY(1)

- **RUNWAY1番を2UNITS確保できるかチェックする．**
- **確保できるならその領域を占有する．**
- **確保できなければ，RUNWAY1番の前で，待ち行列にはいる．**

relinquish 2 RUNWAY(1)

- **占有しているRUNWAY1番を2UNITS開放する．**
- **もし，待っているものがいたら，次のを入れる．**

なお，

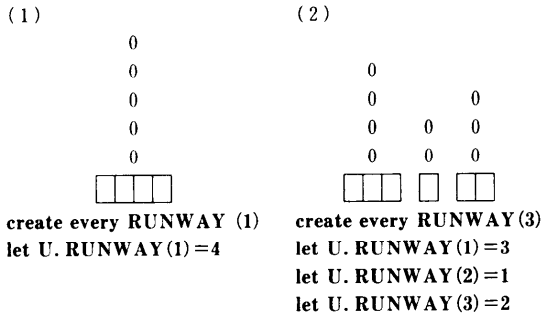
create every RUNWAY(3) ⟷ (同等) **let N. RUNWAY=3**, **create every RUNWAY**

である．

resourceがcreateされると，permanent entityと同様，そのためのメモリが確保される．

その前にできる待行列が，multi-queue, multi-server になるか，single-queue, multi-server になるかを次の例に示す．

(例)



(1)は大きさ4の RUNWAY を1つ create したので，その前にできる待行列はは1つ，すなわち single-queue, multi-server になり，(2)は，大きさがそれぞれ3, 1, 2の RUNWAY を3つ create したので，multi-queue, multi-server になる．

request には，優先順位をつけることができる．たとえば，

Request 1 RUNWAY(2) with priority N

とすると，この request は優先度 N(N は定数，変数，数式いずれでもよい)が付され，N の値が大きい順に request が出される．

work, wait: process を一定時間停止させる，すなわち時間を消費させる命令である．

suspend: process を自身で停止させる命令である．

reactivate: 特定の process を再起動させる命令である．

suspend は，内部処理的にいうと，当該 process の process notice を実行(execute)状態から，create された状態に戻すことにほかならない．ただし，この process notice は，reactivate もしくは resume(後述)により再起動かけられるまで，create された状態で保持される．

interrupt: ある process に割込みをかける命令である．

resume: 割込みを解除する命令である．

interrupt 命令は

INTERRUPT〔THE〕process〔CALLED variable〕

の形で出される．割込みをかけられる process は，work もしくは wait で時間消費(pending)中であることが前提である．割込みをかけられた時点であとどれくらいの時間消費が残っているかが，当該 process の attribute, TIME. A に蓄えられる．後で process が resume されたとき，TIME. A の時間だけさらに時間消費することになる．TIME. A の値は，割込みを受けている間に修正可能である．

resume 命令で割込みを解除された process は pending 状態に戻され，TIME. A で示される時間経過の後活性化される．

- **create**: ある process を動かすための実態を確保する命令である．
- **destroy**: create されている process の実態を消滅させる命令である．

create, destroy 命令は temporary entity に対しても用いられる命令であるが，process もある種の temporary entity と見なして同様の処理を受ける．

より詳しくいうと，create は，ある形の process の process notice のためのメモリを確保する．この process notice は，event list にはおかれない．activate 命令には，create 機能が含まれている．process を起動させる前に，あらかじめその attribute に値をセットしたいときに create が使われ，後で activate される．attribute を用いず，process の引数を使用するときは直接 activate する．

destroy は逆に create された process notice のためのメモリを開放する命令である．create されるばかりで，destroy がなされないと，そのためのメモリは増加する一方となる．

8. process の状態の変遷

SIMSCRIPT II.5 で process の状態がどのように変遷するかは，比較的とらえにくいのでこれを図示し，説明を加えておく．

まず**図-2**は，外からの命令で process の状態がどのように変遷するかを示すものである．

最初は，システムに何も存在しない状態(非存在状態)であるが，create 命令で，process の処理のための実態(メモリの割当て)が確保され，create 状態に移る．create 状態の process が

destroy されると，非存在状態に戻る．create された特定の process(the process)は，ほかの process からの activate により，run 状態(active)に移る．時に attribute のセットなどが必要でない場合は，直接他の process からの activate により，非存在状態から run 状態に移る．

run 状態の process に対して，外から割込み(interrupt)がかかると，非活動状態の create された状態に移るが，resume もしくは(re)activate により再び run 状態に戻る．

process がある resource について request を出したとき，これが使用可能であれば，そのまま active 状態にとどまるが，ほかの process がすでに同じ resource を使用中の場合は，resource 待ち状態に入る．ある時間が経過して，かねてその resource を使用中であった process が，その resource を relinquish すると，resource 待ち状態であった process が run 状態に戻る．

なお，**図-2** の実線はただちに行われる行動，点線はある時間経過後の行動を示す．

1 方**図-3** は，ある process がみずからの出す命令で，どのような状態の変遷を見せるかを示すものである．

まず当該 process が run(active)状態にあるとして，work もしくは wait 命令でいったん pending 状態に入っても，所定の時間が経過すると，run 状態に戻る．その process が run 状態のとき，suspend 命令でみずからを停止させると，run 状態から create された状態に移行する．またある resource を request したとき，これがほかの process に使用されていて使用できないときは resource 待ち状態に入る．また当該 process は return もしくは end 命令で，非存在状態に戻される．

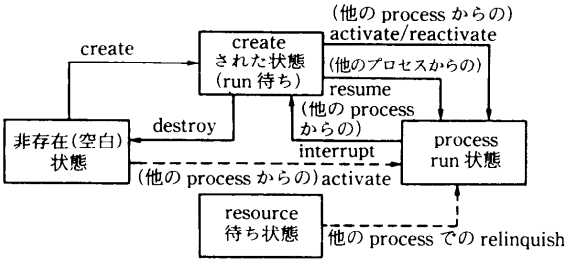


図-2 process の状態遷移(ほかから受ける処理)

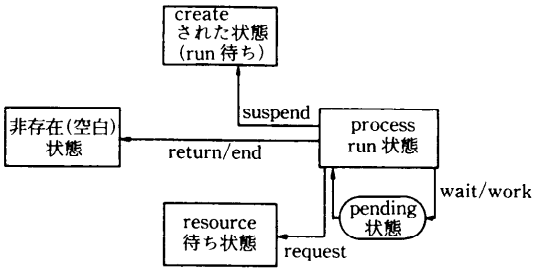


図-3 process の状態遷移(みずから行う処理)

なお SIMSCRIPT II.5 のプログラミング機能としては，以上のほか，確率過程の表現，統計情報収集方法などがあるが，紙面の都合上割愛した．

9. SIMANIMATION[4]

SIMANIMATION は，SIMSCRIPT II.5 で使用できるアニメーション用の拡張機能である．この機能は，モデルの動きを時間経過に沿って視覚的に捉えることができるようにしたもので，大別すると PRESENTATION GRAPHICS (**図-4**)と ANIMATED GRAPHICS(**図-5**)の 2 種類がある．

(1) PRESENTATION GRAPHICS

任意オブジェクトの属性について，時間経過とともに刻々と変化する様子をグラフ形式で画面表示させる機能で，そのグラフ表示方法には次の 6 とおりがあり，複数表示も可能である．

- 円グラフ(Pie chart)
- X-Y グラフ(X-Y plots)
- 時間経過表示(Clock)
- レベルメータ(Level meter)

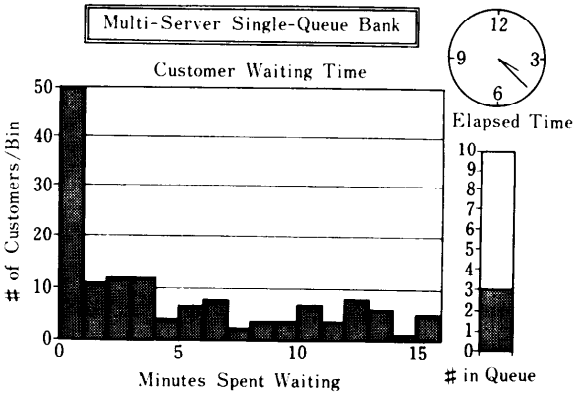


図-4 PRESENTATION GRAPHICS 例

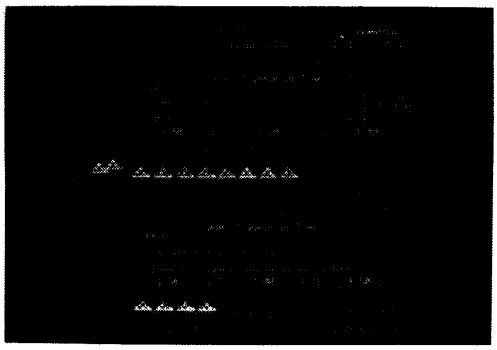


図-5 ANIMATED GRAPHICS 例

●ダイアル(Dial)

●ヒストグラム(Histogram)

実際には，プログラムの宣言部(PREAMBLE)の統計情報命令文(TALLY, ACCUMULATE)で指定するだけで画面表示が可能となる．

(2) ANIMATED GRAPHICS

オブジェクトをアイコンで定義し，そのアイコンを画面上で動かす(プログラムする)ことによって視覚的にシステム全体を把握できる．このことによって，矛盾点やネックとなる個所を検出することが可能となる．

まず，アイコンエディタによってオブジェクトの概形を作画する．実際には，標準的アイコンの使用も可能である．次に，オブジェクトの動きを設定する PROCESS をプログラムで作成する．このプログラムの中に，アイコンの動く軌跡，速度，加速度を記述することになる．

SIMSCRIPT はプログラミング言語であるため，SIMANIMATION としての動画を記述するには，SIMSCRIPT のプログラムの中にアニメーションプログラム部分を何カ所かに分けて挿入することが必要となる．したがって，それ相応のプログラミング技術は必要ではあるが，複雑なシステムの表現も可能である．

図-5 は，金鉱山における金鉱石搬出のシミュレーションである．エレベータに待ち行列に並んでいる金鉱石をのせ，地上へ運んでいる．ここではエレベータや金鉱石がアニメーションとしてダイナミックに動き，時間経過による待ち行列の長さがグラフとして出てきている．

10. おわりに

SIMSCRIPT II.5 は汎用言語的性格をも持っている関係で，言語仕様，モデル構成概念いずれも難解で，初心者にも分かりやすい解説を書くのは至難のことと思われる．特に割込み過程は GPSS と異なり，permanent entity(GPSS では facility)使用へのそれではなく，resource を使っている process への割込みの形をとっている．

紙面の都合上舌足らずの解説となったが，実際に SIMSCRIPT II.5 でシミュレーションを行いたい方々は，ぜひ後掲の文献を参照されることをおすすめする．

参考文献

1) Russell, E. C.: Building Simulation Models with SIMSCRIPT II.5, CACI(Sep. 1983).
2) Mullarney, A.: SIMSCRIPT II.5 Programing Language, CACI (Jan. 1983).
3) 中西俊男：シミュレーション，電子情報通信学会編，コロナ社(July 1994).
4) CACI: SIMGRAPHICS User's Guide and Casebook (Jan. 1991).

(平成7年2月27日受付)

中西 俊男(正会員)

1932年生．1956年東京大学理学部数学科卒業．日本国有鉄道，鉄道技術研究所勤務．1969年工学博士(東京大学)．1972年成蹊大学工学部経営工学科教授．主たる研究テーマ：道路交通管制システムシミュレータなど各種シミュレーションシステム，情報管理システムの開発．著書「コンピュータシミュレーション」，「コンピュータの基礎」，「点と線と面」，「シミュレーション」，他．日本シミュレーション学会理事，SCS，OA 学会，日本経営工学会，他各会員．