Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

SIMULATION WITH PCModel

Young G. Chung
David A. White
Simulation Software Systems, Inc.
San Jose, CA 95131

ABSTRACT

This paper reviews a PC-based graphic simulation system known as PCModel. PCModel takes a unique modeling approach by providing the user with a very simple yet powerful set of instructions for graphic modeling plus a highly interactive run-time control commands for effective What-if gaming. The flexibility and user-interactiveness of this system provides an accurate and economical tool for analyzing manufacturing systems.

1. INTRODUCTION

PCModel is a graphic simulation system specifically designed to model the movement and decision making of OBJECTs as they flow through a series of process steps called a ROUTING. Typical objects would be machine parts, silicon wafers, purchase orders or invoices, electronic messages, appliances, printed circuit boards, or pallets.

PCModel has a unique feature of displaying the actual movements of the objects in simulated time. A crucial feature of PCModel is that only one object can occupy an overlay location at a time. The physical contingency of objects, parts, or resources is automatically sensed and maintained by the underlying system.

PCModel will function on the IBM PC, PS/2, or AT (and compatibles) with no additional hardware requirements.

2. PCMODEL FEATURES

PCModel provides the user with a main simulation screen, seven dynamic information screens, and a variety of run-time commands and modeling instructions. This is what makes PCModel unique, because this enables the user to have a complete control over the simulation while the model is running as well as to develop a sophisticated model using the modeling instructions.

The PCModel system automatically updates and maintains the dynamic information screens. Any of the these screens can be accessed by the user during the simulation to provide up-to-date information on the model status. The Help Menu screen of PCModel shows all the run time commands available for controlling the simulation. The following diagram shows the organization of PCModel.

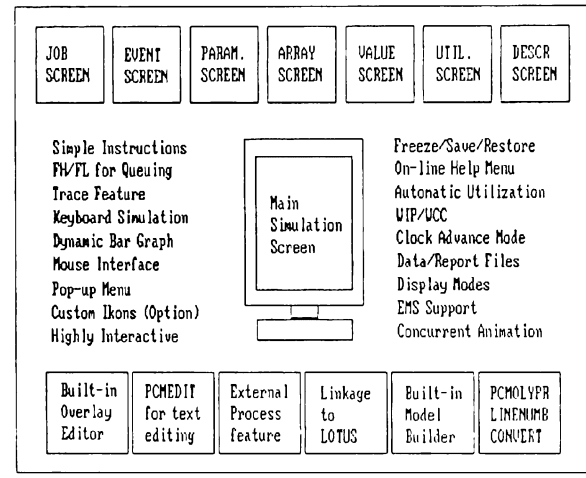

Figure 1: Organization of PCModel

Other features of PCModel are summarized as follow;

- You can **freeze** the simulation by pressing the space bar and resume it with G key. You can **save** the simulation in progress to a disk and **restore** the saved simulation at any time.
- You can define up to 100 locations for **automatic collection of percentage utilization**. The utilization screen can be accessed at any time during the simulation.
- A **help menu screen** can be accessed by pressing H or F1 key without disturbing the running model.

- A **diagnostic trace feature** allows you to trace object statement processing on a line-by-line basis.
- PCModel typically simulates at **greater than 1000 times** real time. The simulation pace can be adjusted.
- **Work-In-Process** and **Work-Complete-Counts** are constantly displayed.
- You can display statistics with **dynamic bar graphs.**
- You can easily create your own **pop-up menus.**
- You can select **different display modes.**
- A pop-up **system status screen** is available to show the current status of memory allocations.
- You can read and write free format ASCII **data/report files.**
- You can create sound effects.

The PCModel system also comes with supporting utilities that make modeling even easier.

- The built-in **Overlay Editor** allows the user to define the overlay over which the objects (entities) will be traveling. It features text edit, line and box drawing, fill and copy functions as well as macro image save and retrieval capability (The complete IBM character set is supported in sixteen colors). The **logical screen** can be over 32,000 grid coordinates and the ten number keys can be assigned to different views of a model overlay larger than the display screen.
- The **eXternal Process feature** of PCModel supports interaction of the model with user written programs in C, Fortran, or Pascal. This feature allows a variety of new applications, such as the real-time process monitoring, calculation of complicated mathematical equations, and so on. This feature along with the EMS (Expanded Memory Specification) support enables modeling of very large and sophisticated models.
- Software is available to assist pointing and menu selection using a **mouse or other pointing devices.**
- A software utility is available to create your own **custom characters** to use as ikons for adding realism to object representation.
- The utility PCMLOTUS converts statistics files into a format acceptable by Lotus. The utility PCMLOTUS is a Lotus worksheet used to calculate the Mean, Standard deviation, Minimum and Maximum percentage utilization for any specified period.
- The utility PCMOLYPR converts an overlay file into a format that can be printed. LINENUMB creates a line-numbered version of the application file for easy debugging.
- The **Builder** enables an inexperienced modeler to build workstations/transporter models graphically without writing any code.
- **PCMEDIT**, the supplied full screen text editor, allows the user to edit programs.

3. MODELING PROCEDURE WITH PCMODEL

In general, development of a successful simulation model requires the model developer to follow a certain procedure. There may be slight variations depending on the software. The first step in model development is to define the problem and prepare a model specification. This provides a basis for designing the model structure and data input and output formats. Once a specification is ready, data collection and model building can be started. As long as all the input parameters and their formats are defined in the specification, you can create a dummy input data file and start the model building activity.

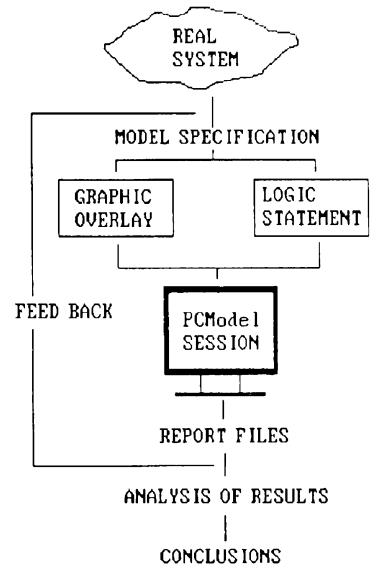

Figure 2: Modeling Procedure

Building a simulation model with PCModel involves two components. You need to draw an overlay, a schematic representation of the actual system, using the built-in graphic editor. Then you create a model statements that describe the

movement and behavior of the system components, i.e. entities. Preparation of both the overlay and model statements can be done without ever leaving the PCModel system, since you can invoke the overlay editor and the text editor from the PCModel help menu screen.

Once the model is ready (both overlay and model statements), you can analyze the system behavior as the model runs and try What-If scenarios. You can also create report files after each simulation run for further statistical analysis.

If any change needs to be made as you verify and validate the model, you can invoke either the overlay editor or the text editor to make the appropriate change anytime during simulation. You can then reload the model and start a new simulation run without leaving the PCModel system. This allows the user to develop a complete model in an extremely efficient fashion. The modeling procedure using PCModel is summarized in figure 2.

4. PROTOTYPE MODEL FILE STRUCTURE

The typical structure of a PCModel application (statements) file can be viewed as consisting of two major parts: load-time directives & symbol definition, and run-time instructions.

Load-time directives & symbol definitions consist of directives that are executed only during the loading process. They initialize and direct the simulation environment. Examples include specification of the background Overlay file, reservation of memory space for symbols, definition of WIP limit, specification of the manner in which groups of objects following the same route are to be released into the model, definition of utilization statistic collection points, and definition of variable names and initial values assignments, definition of array names and dimensions.

Run-time instructions are executed only while the simulation model is running. Instructions are used to control the movement of objects or to update model parameters. The various movement paths and decision making processes that objects follow are called routing and are made up of a sequence of run-time instructions. Links (=subroutines) are routing segments that can be used by multiple routes.

```
;----DIRECTIVES AND SYMBOLS----------------------------------------------------------------------
L=(40)                                  ; Wip Limit to 40 objects
D=
      BARBERSHOP SIMULATION
      ONE-LINE SINGLE-SERVER QUEUEING MODEL
      John Peters, Simulation Software Systems, 10/20/87
$
O=(=)                                   ; Overlay File with the same name
;
%INTER.ARRIV=(15:00)                    ; Mean time between arrivals
%ARRIVAL=(00:00)                        ; Random time of arrivals
%HAIR.CUT=(15:00)                       ; Mean time required for a hair cut
%HAIR.STD=(2:00)                        ; Std. deviation for a hair cut
%CUT=(00:00)                            ; Random time to get a hair cut
@WAIT.CHAIR=(30)                        ; Maximum number of chairs for waiting
*DOOR=(XY(7,15))                        ; Door location on the screen
*BARBER=(XY(65,15))                     ; Barbers position
J=(1,*,1,0,0,0,5000)                    ; Job Directive - Job #1
U=(1,Barber,*BARBER)                    ; Collect the utilization of the barber

;----BEGIN ROUTE NUMBER ONE---------------------------------------------------------------------
BR(1,*DOOR,%ARRIVAL)                    ; Begin Route 1 at *DOOR, Expo. arrival times.
MR(@WAIT.CHAIR,0:00.00)                 ; Move object right @WAIT.CHAIR steps
RV(N,%CUT,%HAIR.CUT,%HAIR.STD)          ; Rand. hair cutting time (Norm. Distr)
MA(*BARBER,%CUT)                        ; Move to barber and get hair cut
ER                                      ; End of Route 1
```

Figure 3: A Barber Shop Model

A complete program listing of a barber shop model is provided in the Figure 3. Directives and variables are defined and initialized in the top portion of the program, then the movement and decision making of the system entities are described in the routing.

5. RUN-TIME INSTRUCTIONS

Begin Route, End Route; A ROUTING is initialized with a **BR** instruction and ended with an **ER** instruction. This **BR** (Begin Route) instruction defines a routing number, where the starting point is to be in the Overlay picture, what delay (hours, minutes, seconds and hundredths of a second) there is before proceeding to the next instruction, as well as establishing the inter-arrival time of objects entering that routing.

BR(route,xy,tt), ER

Move Up, Move Down, Move Right, Move Left, Relative Move; **Move** instructions are either *relative* or *absolute* in their action. Move up, down, right, left instructions move the object a relative distance from where it started, coming into this instruction. **RM** is another relative move instruction which also moves an object a relative distance, but with a slightly different approach. The **MU**, **MD**, **MR**, and **ML** instructions delay movement for the designated amount of time. Then they check that the next location (ie the adjacent, *column-row cell* in the direction of movement) is not occupied before moving to it. They do this for each iteration until reaching their destination. For example, MU(3,%MOVET) moves the object up 3 rows, with a delay of %MOVET after *each* of the 3 iterations. Note that the time delay for all these instructions may in fact be zero, causing the object to move in zero time.

MU(vv,tt), MD(vv,tt)
MR(vv,tt), ML(vv,tt)
RM([vv,]+xx|-xx,+yy|-yy,tt)

Move Absolute; One must use an absolute move instruction, **MA**, when it is not certain where the object is located (such as coming into this instruction from a **jump**).

MA(xy,tt)

Wait Time, Wait Clock, Wait Advance, Wait Event; These instruction cause the object to wait for a certain time delay before it executes the next instruction. **WT** adds a specific time delay you define, whereas **WC** causes the object to wait until the simulation clock becomes equal to the specified time.

WT(tt), WC(tt)

WA causes the object to wait until the simulation clock advances. **WE** causes the object to wait at the current instruction until at least one new event occurs.

WA, WE

Test Position; **TP** is used to test the target screen location(s) for occupancy. If any of the specified locations are blocked or posted, the object will wait until all locations are clear.

TP([A,]xy[,...,xy])
TP(R,(+|-xx,+|-yy)[,..,(+|-xx,+|-yy)])

POst, CLear location; **PO** posts a location as occupied. **CL** is used to clear a posted location. **PO** and **CL** are used in conjunction with **TP** for synchronizing objects movement.

PO(xy[,char[,bg[,fg]]])
CL(xy[,char[,bg[,fg]]])

JumP; This instruction is used to make an unconditional jump to a specified routing label for controlling logic flow. You need to specify the label to jump to.

JP(:c..c)

Jump if Blocked; This is similar to JP, but it jumps to a specified routing label only if any of the specified locations are blocked. Otherwise, the next instruction will be executed. You can specify absolute or relative locations.

JB([A,]xy[,...,xy],:c..c)
JB(R,(+|-xx,+|-yy)[,...,(+|-xx,+|-yy)],:c..c)

Jump if Cleared; This instruction will cause the object movement control to transfer to the label only if ALL locations specified in the instruction are cleared. Otherwise, the next instruction will be executed.

JC([A,]xy[,...,xy],:c..c)
JC(R,(+|-xx,+|-yy)[,...,(+|-xx,+|-yy)],:c..c)

Random Seed; **PCModel** allows for using up to ten different random number streams. This instruction is used to initialize the pseudo-random number generation function by specifying its seed.

RS([stream,]nn)

Random Value; This instruction is used to generate a random number following a certain distribution. **PCModel** supports Normal, Exponential, and Uniform distributions. If your data does not form a known distribution, you can define your own custom distribution (both continuous and descrete). Note that the random number stream should have been initialized with **RS** before **RV** is used for the first time.

RV([stream,]rand,lolimit,hilimit)
RV([stream,]U,rand,mean,spread)
RV([stream,]N,rand,mean,deviation)
RV([stream,]E,rand,mean)
RV([stream,]C,rand,table)
RV([stream,]D,rand,table)

Set Value; This instruction is used to set the value of a symbolic reference to a known value. It allows mixed operations between all data types and object parameters (with a few restrictions).

SV(vv,nn)

Arithmetic Operation; This is used for arithmetic operations on two values. Note that the first operand is multiplied by the second operand and the result is stored in the first operand.

AO(vv,+|-|*|/,nn)

Increment Value, Decrement Value; **IV** increments, and **DV** decrements the present value of a variable by 1. Note that clock values will be incremented or decremented by 00:00:00.01.

IV(vv), DV(vv)

Print Message; **PM** is used to print a message at a specific location on the screen or to a report file. You can specify the background and foreground colors when printing on the screen. When printing to a file, a comma after the message will concatenate the next message.

PM(xy,"msg"[,bg[,fg]])
PM(F,"msg"[,])

Print Value; **PV** is used to print a value at a specific location on the screen or to a report file. You can specify the background and foreground colors when printing on the screen. Also you can control the start and length of the value to be printed. When printing to a file, a comma is used to concatenate the next field.

PV(xy,vv[,start[,length[,bg[,fg]]]])
PV(F,vv[,start[,length]][,]))

IF; This instruction is used to compare two different values and select the processing statement based on the result.

IF(nn,cond,nn,[THEN][GOTO],true[,ELSE][,GOTO][,false])

Begin Link, End Link, LinK; This instruction is used to specify the start of a link with a link name. A link in **PCModel** is equivalent to a subroutine in Fortran. Use **EL** to designate the end of a link. **LK** causes the object to jump to the specified link and execute from the first instruction of the link. At the end of the link, the object will jump back to the next instruction after the **LK** instruction.

BL(!label), EL
LK(!label)

Initialize Array; This instruction initializes all members or specified column or row elements of an array to a specified value. Default value is zero.

IA(array[(col,row)][,value[,start[,length]]])

Set Color; This instruction allows you to control the foreground and background colors of an object.

SC(bg,fg)

Find High, Find Low; These instructions are used to scan a specified column or row of an array to find its highest/lowest value, returning the index (or indices) to the value. You can also specify a second array to be used as indices into the array of values.

FH(highest,data(c,r)[,index(c,r)][,start[,length]])
FL(lowest,data(c,r)[,index(c,r)][,start[,length]])

Get Character; This instruction is used to get the character ID (ASCII value) and color attribute value for any character displayed on the overlay.

GC(xy,char[,bg[,fg]])

Generate Sound; This instruction is used to generate a sound. You can control the duration and the pitch of the sound.

GS([pitch[,duration]])

Get Value; This instruction is used to accept a value from the user.

GV([xy,]value)

Time Stamp; This instruction is used to access the the actual time used by your system.

TS(%var)

KeyBoard; This instruction basically allows you to simulate any keyboard action from within your program.

KB(command)

Simulate Event; This instruction is used for signaling to the system that something has happened to change the state of the simulation, which normally occurs for all event type instructions.

SE[(I)]

6. APPLICATIONS OF PCMODEL

PCModel has been used by more than 1200 professional engineers world-wide. Its application areas include Production and Assembly Line, Warehouse Operation, Material Handling Systems, Job-Shop Scheduling, IC Fab Operation, Service Industries, and JIT/Kanban System, to list a few. The following presents some of the PCModel applications with a brief description.

Group Technology - This model is developed to compare performances of different process layouts in a bike assembly plant; the conventional process layout by product and a new design with GT concept. This model clearly demonstrates the concepts and benefits of Group Technology.

Circuit Card Assembly Line - This is a simulation of an electronic circuit card packaging facility consisting of an automatic transportation system and four types of insertion machines equipped with automatic load-unload devices. This model shows dynamic queue status and statistics using bar graphs. Different process routings and device dependent process times are specified in a user input data file.

Automated Kitting Facility with Carousel Inventory - This model was developed to simulate a kitting process of military communication equipments at Rockwell International, Dallas, TX. The primary purpose of the model was to verify their proposed system design against throughput objectives. It also had a secondary purpose of exploring various proposed control algorithms to be implemented as part of the final process-control software.

Integrated Circuit Fabrication Processing - The model was developed to graphically simulate the fabrication activity at a major I.C. manufacturer. The model has proved to be an excellent vehicle for evaluation of scheduling rules, for studying the impact of changes in machine and operation attributes on system performance, and in training managerial personnel.

Random Access Wafer Processing Equipment - This model was developed for an manufacturer of wafer processing equipment, and provides a tool for designing control algorithms for wafer transfer system and for evaluating different equipment layout tradeoffs.

Warehouse Simulation - This model was developed for a major garment industry distributor implementing radio-dispatched material handling equipment. The primary objectives of the model were to analyze material flow patterns in the warehouse and to evaluate different dispatching rules for radio equipped trucks.

Containerized Marine Cargo Facility - This model was developed to study the impact of container layout strategies in the container-yard on the utilization of the material handling equipment and the efficiency of the ship loading process.

7. CONCLUSIONS

PCModel provides the modeler with a complete environment for efficient modeling activities. Unlike other menu-driven simulation systems, PCModel requires the modeler to be fluent with a set of high level programming instructions. This tradeoff results in more detailed and flexible simulation models that more accurately reflect the actual or proposed systems.

8. REFERENCES

Chung, Y. G. (1986). An Animated Simulation Model for a Transtainer-based Container Handling Facility. Master's Thesis, Oregon State University, Corvallis, Oregon.

White, D. A. (1989). PCModel, Personal Computer Character Graphic Modeling System. Simulation Software Systems, Inc., San Jose, California.

Chung, Y. G. (1988). Course Outline for Graphic Simulation using PCModel and CADmotion. Simulation Software Systems, Inc., San Jose, California.

9. AUTHOR'S BIOGRAPHY

Young G. Chung is a simulation analyst at SimSoft. He has been providing graphic modeling and engineering consulting services for a variety of industries since 1986. He holds a Bachelor of Science in Industrial Engineering from Hanyang University and a Master of Science in Industrial Engineering from Oregon State University.

Young G. Chung
Simulation Software Systems, Inc.
2107 North First Street, Suite 680
San Jose, CA 95131
(408)436-8300

David A. White is president and founder of Simulation Software Systems, Inc. He received his B.S. in Electrical Engineering from San Jose State University in 1974. Prior to founding his company, he worked as an Advisory Manufacturing Engineer for IBM's General Product Division. In that capacity he had technical responsibility for establishing manufacturing and test processes for that company's strategic products. His current activities include enhancement of personal computer based modeling and simulation software and techniques. He is a member of SCS and IIE.

David A. White
Simulation Software Systems, Inc.
2107 North First Street, Suite 680
San Jose, CA 95131
(408)436-8300