ACTION - A GRAPHICS AID TO INTERACTING WITH MODELS AND SIMULATIONS

P.P. Tanner and K.B. Evans
National Research Council Canada

ABSTRACT

ACTION is a programming system that adds interaction capabilities to existing computer models and simulations. While the model is running, a vector graphics display gives a dynamic pictorial representation of selected simulation variables. The user, while viewing this display, may modify model parameters using input control devices to interact with the model.

ACTION - INTERACTION AVEC DES MODÈLES ET DES SIMULATIONS AIDÉE PAR L'INFOGRAPHIE

RÉSUMÉ

ACTION est un système de programmation qui permet d'ajouter des possibilités d'interaction aux modèles d'ordinateur et aux simulations déjà existants. Pendant la calculation du programme-modèle, un visuel à balayage cavalier représente graphiquement des variables choisis de la simulation. L'usager examine le visuel et en même temps peut changer des paramètres du modèle avec les contrôles interactifs.

1. Introduction

Computer models and simulations have long been recognized as some of the most useful applications of computers in the research and development field. For specialized applications, interactive modes of operation using cathode ray tubes (CRT's) to display information and interactive consoles to enable the user to control the model have been used extensively. Many dramatic advances in graphics display technology have resulted from the necessity of displaying the results of simultions and models.

Although some models represent physical entities and make use of graphical data bases, most models do not represent physical objects and contain no graphical data base. Graphics are not often used with these models except to produce graphs and histograms showing the final results. Nevertheless, a meaningful pictorial representation of model dynamics offers a powerful tool for interpreting or examining various aspects of model performance. In a simple example, pictures of different sized oil wells distributed on a map could represent relative quantities of oil production in an energy model. The resulting changes in oil production could be observed dynamically by the viewer in response to changes in model parameters such as price, import taxes or demands, all controlled through user interaction.

Adding clear, illustrative graphics to models is a demanding task. Not only must the modeller be familiar with the graphics routine available to him, but he must either be skilled in the user of abstract representations of data, or be willing to experiment to find effective representations. To modify the graphical output of a model, the modeller must often change his model and his graphical data base - a time consuming procedure.

One approach to the problem is to maintain complete separation between the model and its graphical representation. Using the general purpose graphics modelling system described below, a graphical data base and a linkage between the data base and the model may be created and easily modified. The graphics/model linkage allows the choice of a wide range of pictorial representations for any given model.

2. ACTION Programming System

The ACTION programming system (Fig.1) has been designed to facilitate the addition of user-model interaction to existing models. A "master picture" is used to provide the graphical data base, and a program in a special purpose language, ACTION, is used to specify connections between the model and that data base. The ACTION program also describes links between a control console and the

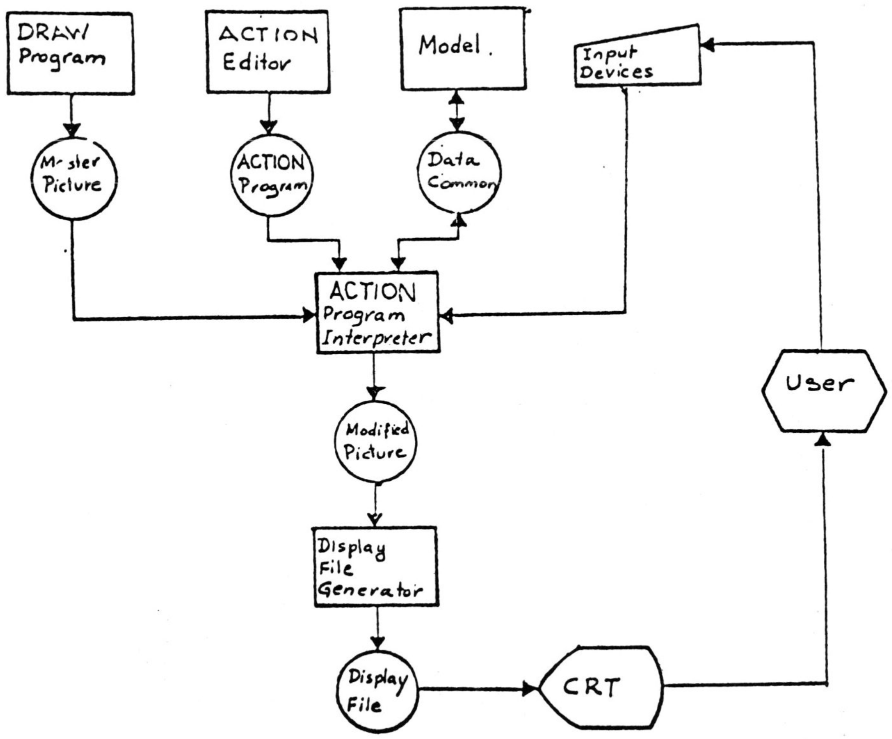

Figure 1. ACTION System Data Flow

model. The facility allows quick modification to the graphical data base or the ACTION program.

When the user has produced the picture and the control program he can run the model. The ACTION programming system will show changes in the model on the CRT and allow the user to control the model in the manner specified in the ACTION program.

The ACTION programming system can be divided into two groups of functions. Those in the first group are performed during the preparation stage prior to a model run, and those in the second, during the model run.

2.1 Preparation

Before a simulation is run, a master picture, an ACTION program, and a model must be prepared. The master image and the ACTION program are prepared using facilities provided by the ACTION programming system; the model is prepared separately. After any run of the model the master picture or the ACTION program may be modified in order to investigate different properties of the model or to experiment with a different pictorial representation of the same properties. Since the ACTION language is interpreted rather than compiled, the model may be run immediately after a modification to the master picture or the ACTION program without a time consuming compilation.

2.1.1 DRAW program

The master picture is drawn with the aid of a graphics tablet. This picture is composed of a number of picture segments - a segment is the smallest portion of the picture that may be modified during a run. Designed to produce segmented pictures, the DRAW program supports both free-hand and geometrical constructions, including tablet controlled arcs, gussets, lines restricted to specific lengths or angles, endpoints restricted to a reference grid, as well as transformation facilities. Picture segments may be duplicated, and pre-defined geometrical shapes may be added to the picture. This facility uses an interactive graphical dialogue technique (described below) that is common throughout the system.

2.1.2 ACTION program

The ACTION program, written and/or modified during the preparation stage, becomes the controller of the running stage. Run once for each frame update on the CRT, it acts as a switchboard, linking simulation variables to picture segments, connecting input devices to other variables, and translating simulation variables into text for CRT display.

2.1.3 Simulation or Model

The simulation or model program must be written in FORTRAN or other language that runs under the PDP 11's RSX 11M operating system. All internal variables used to control picture elements, or variables that the user may wish to control during a run, must be placed in a data common. The model must have two program entries, one called before the model run for initialization, and a second called for each frame.

2.2 Running Stage

When the simulation is running, several programs are used to produce each update of the image on the screen. Fig. 2 shows the control loop - the sequence in which these programs are executed, and Fig. 1 shows the data flow amongst the programs. The first program in the loop restores the graphical data base to the original "master picture". Then the ACTION program is executed by the interpreter. The picture restoration simplifies the writing of the ACTION program by eliminating the accumulation of the transformations from frame to frame, and eliminating the build-up of round-off errors.

The ACTION program uses values stored in the data common to modify the master image. It also places specified input device readings into the data common. When the ACTION

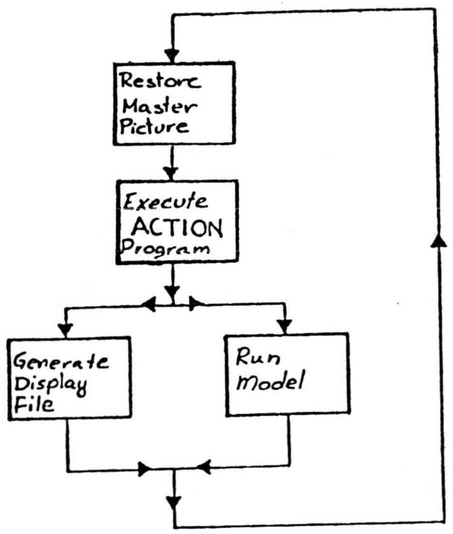

Figure 2. ACTION System Control Flow

program has been executed, the display file with the new picture, and the model is called to provide data for the next frame.

3. ACTION Language

The design of the language was based on the following assumptions: the length of the average ACTION program would be less than 50 lines; it would rely heavily on special purpose graphical functions; it would be mainly linear in structure; and it would make limited use of conditional statements or iterations. (Of course the whole program is interactive in the sense that it is executed once for each picture frame). This has led to a language that has many graphical functions, but has a restricted set of control statements.

The ACTION language is similar in many ways to DeFanti's GRASS language (2), a language directed towards "performance graphics" and animation. However, the divergence in aims of the two languages has led to major differences in both the supporting programming systems and the manner in which the programs are executed (6).

The language is described here using an example of a spring-mass model, a picture of a spring and a mass, and the corresponding ACTION instructions that could be used to animate the model. Fig. 3 gives the FORTRAN program modelling the motion of a spring and mass. The program has two entries: MODINI initializes the constants and is called automatically by the ACTION system at the start of any run; and the entry MODEL is the actual model called once each frame. The model uses values provided by the ACTION program (XINPUT,YINPUT) to control the supported end of the spring (XTOP, YTOP) and computes the position of the

```
C
C   SPRING MASS MODEL
C
       COMMON/GLOBAL/VELX,VELY,XMASS,YMASS,XTOP,YTOP,XINPUT,YINPUT
      1              SPRCON,DAMP,TIME
C
C   VARIABLE NAMES:
C   (VELX,VELY) - VELOCITY IN X AND DIRECTIONS
C   (XTOP,YTOP) - POSITION OF TOP OF SPRING
C                 (RELATIVE TO POSITION IN MASTER IMAGE)
C   (XMASS,YMASS) - POSITION OF BOTTOM OF SPRING
C                 (RELATIVE TO POSITION IN MASTER IMAGE)
C   (XINPUT,YINPUT) - POSITION CONTROL FOR TOP OF SPRING
C                     (SET BY ACTION PROGRAM)
C   SPRCON - SPRING CONSTANT
C   DAMP - DAMPING FACTOR
C   TIME - TIME BETWEEN CALLS TO MODEL
C
C   INITIALIZE ROUTINE
       ENTRY MODINI
       SPRCON=.995
       DAMP=.985
       XMASS=0.0
       YMASS=0.0
       VELX=0.0
       VELY=0.0
       TIME=.25
       RETURN
C
C   ACTUAL MODEL
C
    ENTRY MODEL
C
C   POSITION OF TOP OF SPRING IS SET BY X,Y INPUT
C
       XTOP=XINPUT
       YTOP=YINPUT
C
C   CHANGE IN VELOCITY=-KX
C
       VELX=-SPRCON*(XTOP-XMASS) + VELX
       VELY=-SPRCON*(YTOP-YMASS) + VELY
C
C   NEW POSITION OF MASS
C
       XMASS=XMASS-VELX*TIME
       YMASS=YMASS-VELY*TIME
       RETURN
       END
```

Figure 3. Spring Mass Model

weighted end of the spring (XMASS,YMASS).

To illustrate the model, we draw a master picture of the spring and mass as shown in Fig. 4. It is composed of three picture segments: the TOP support of the spring, the SPRING itself, and the MASS at the bottom of the spring. A basic ACTION program could be:

```
100   COMMON VELX,VELY,XMASS,YMASS,XTOP,YTOP,XINPUT,YINPUT
200   COMMON SPRCON,DAMP,TIME
300   XINPUT=TABLET_X
400   YINPUT=TABLET_Y
```

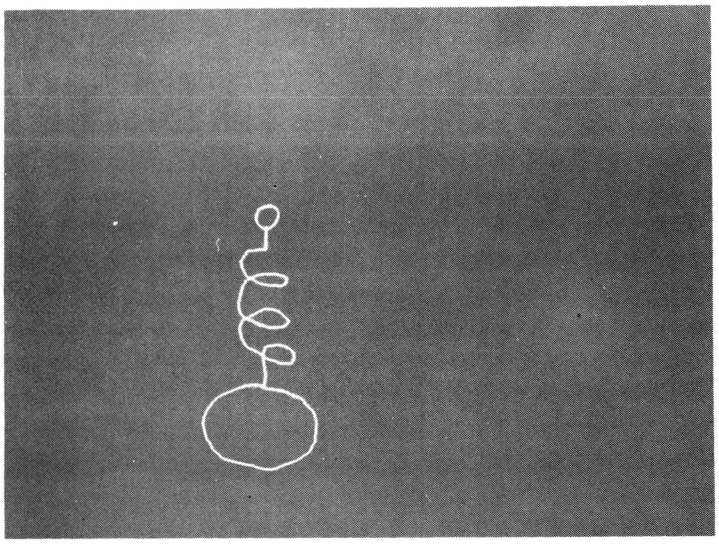

Figure 4. Master Image.

```
500  POSITION_XY(TOP:SPRING,XTOP,YTOP)
600  POSITION_XY(MASS,XMASS,YMASS)
```

This program introduces three types of ACTION statements; the COMMON, the assignment statement, and the stand-alone function. The COMMON statements establish the equivalences of variables used in the ACTION program to variables in the model. The two assignment statements 300 and 400 use the key words TABLET_X and TABLET_Y to send tablet values to the model. Stand-alone functions, lines 500 and 600 are the most common type of ACTION statement and ordinarily use a real variable to control one or more picture segments. For example, line 600 uses the model variables XMASS and YMASS to control the X and Y positions, respectively, of the picture segment MASS.

Line 600 introduces two types of variables: picture segments, defined when drawing the picture, and real values. A real value can be a model variable, a local variable, an input device value, an arithmetic function, or an arithmetic expression of any of these.

Line 500 is similar to line 600 but instead of a segment variable, a "segment set" variable is used, indicating that the operation is to be performed on each segment in the set {TOP,SPRING}.

This ACTION program places tablet information in the data common for the next iteration of the model and uses results received from the previous iteration of the model to control the current position of the three segments. When the program is run the user will notice that moving the tablet pen across the tablet will cause the TOP and the SPRING to move together, the MASS will bounce around as if connected by a spring, however, the SPRING will remain vertical and unconnected to the mass (Fig. 5)

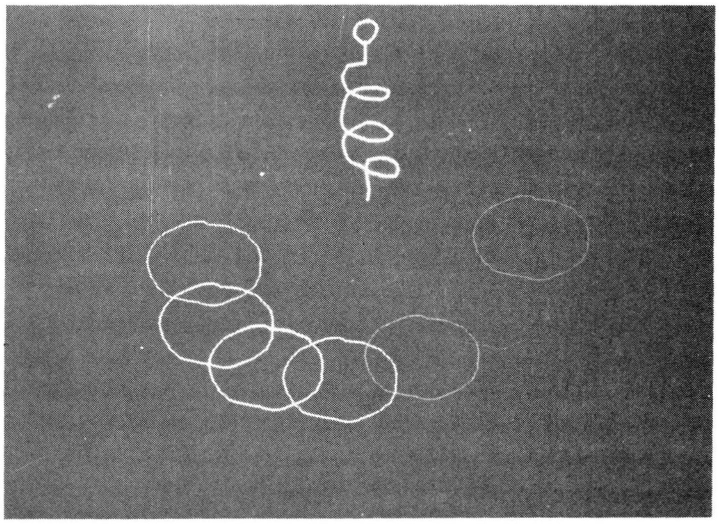

Figure 5. Image during run of first ACTION program.

To make the SPRING appear connected to both the TOP and the MASS, it must be stretched to a length equal to the distance between the TOP and the MASS, and rotated to the proper angle. Since the SPRING is already connected to the TOP neither operation may move the TOP end of the SPRING. The modified ACTION program below improves the animation of the spring mass model and introduces additional ACTION language concepts.

```
100  COMMON VELX,VELY,XMASS,YMASS,XTOP,YTOP,XINPUT,YINPUT
200  COMMON SPRCON,DAMP,TIME
204  DECLARE SPRLGH,SPRSCL,SROT
208  SRPLGH=LENGTH(POINT(TOP,1),POINT(MASS,1))
300  XINPUT=TABLET_X
400  YINPUT=TABLET_Y
500  POSITION_XY(TOP:SPRING,XTOP,YTOP)
600  POSITION_XY(MASS,XMASS,YMASS)
700  SPRSCL←LENGTH(POINT(TOP,1),POINT(MASS,1))/SPRLGH
800  SCALE_Y(SPRING,POINT(SPRING,1),SPRSCL)
900  SROT←H_ANGLE(POINT(TOP,1),POINT(MASS,1))+1.57
1000 ROTATE_(SPRING,POINT(SPRING,1),SROT)
```

Point variables are used extensively in this program. A point variable is an ordered pair of real values, and is usually specified in one of two ways. The expression (100, 150) specifies a static point on the screen at position X=100, Y=150. POINT(SPRING,N) is a function that returns an ordered pair of real values representing the current position of the Nth point of the picture segment

SPRING. An example of point variable use is in line 100 where a point function specifies the centre of a rotation. The real variable SROT controls the rotation of the picture segment SPRING about the first drawn point in SPRING - the point where the SPRING connects with the TOP.

The ACTION language also has functions that extract information from the picture for use by subsequent ACTION statements. For example line 208 returns a real value equal to the distance between the first point of picture segment TOP and the first point in picture segment MASS (where the SPRING and MASS join). In the program, this function is used a second time in line 700. In line 204 the function determines the length of the SPRING in the master image (which is restored just prior to the running of the ACTION program). In line 700 the function calculates the distance between the TOP and the MASS after they have been repositioned. This is the length to which the SPRING will be stretched. Dividing this length by the first gives the scale factor which, when applied to the SPRING in statement 800 stretches it to the correct length. The second parameter in the SCALE_Y function, line 800, is a POINT variable that specifies the origin of the scaling operation.

H_ANGLE is another function that takes information from the picture, namely the angle between a line (specified by two end points) and the horizontal. Statement 900 adds 1.57(=PI/2) to this. The result is an angle relative to the vertical, and in this case, the desired angle of the SPRING. The resulting value is then used to control the ROTATE statement.

Note that statement ordering is important in this program since functions are additive. If ROTATE had preceeded SCALE_Y, the rotated SPRING would have been scaled vertically, producing an interesting but not the intended result.

When the program is run, the user will find that moving the tablet pen over the tablet still moves the unweighted end of the spring, but now the SPRING stretches, contracts, and rotates in a realistic way, and the weighted end of the spring stays attached to the MASS. (Fig. 6)

To investigate the properties of the model in more detail, more statements are added.

```
1100  DAMP←POT_(1)
1200  SPRCON POT_(2)*2.0
1300  PRINT_VALUE(400,400,SPRCON)
```

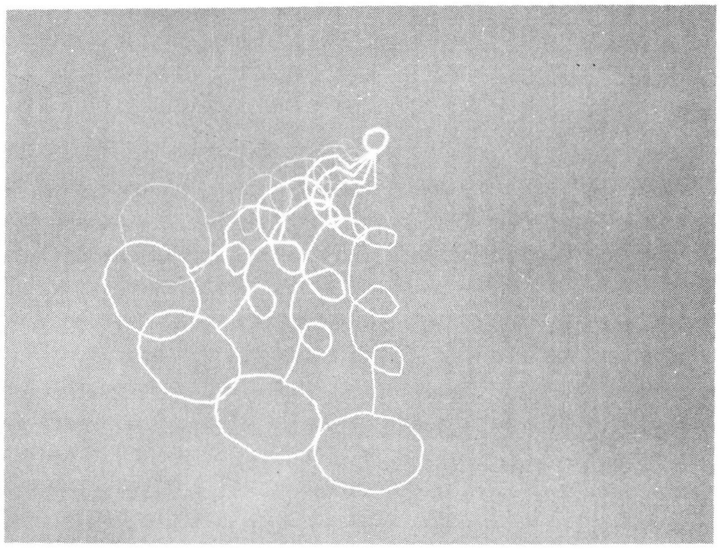

Figure 6. Image during run of second ACTION program

Statements 1100 and 1200 use potentiometers 1 and 2 (range 0.0 to 1.0) to control the model variables DAMP (damping) and SPRCON (spring constant) respectively. Statement 1300 is a function that adds a new picture segment to the image. This function writes the current value of the real variables, SPRCON, at the given point of the screen (400,400). In this case the user can see both the effect of modifying the spring constant on the spring and the actual value of the constant.

In addition to the types of statements described above, the language includes graphical declaration statements and control statements as illustrated in this program.

```
100  COMMON XPOS(6)*YPOS(6)
200  COPY CAR(6)
300  DECLARE I
400  FOR I= 1 to 6
500  POSITION_XY(CAR(I),XPOS(I),YPOS(I))
600  NEXT I
```

Line 200 is a graphic declaration statement that causes five additional copies of the picture segment CAR to be produced. Lines 400-600 define an iterative FOR loop using the syntax of the language BASIC. Each execution of this loop uses position information from the arrays XPOS and YPOS to position one of the copies of CAR. The loop is of course executed 6 times and respositions the six cars.

The ACTION language also contains a large number of graphical functions not illustrated in the above examples. Documentation of these can be found in (4) and (6).

4. Interactive Graphics Dialogue

The ACTION language is an "Intrinsically Graphical Language", "A language in which the program text and graphical objects are treated on an equal footing." [2] It is also a language in which the coding is done by typing text, by drawing pictures and by engaging in an interactive graphical dialogue. The dialogue resembles a question and answer period in which the system not only asks the questions, but also indicates to the user the possible answers.

To illustrate the dialogue, let us assume we wish to input the SCALE_Y statement from the above ACTION program (line 800). On choosing an editor command to input code, the user is presented with a menu of statement types including stand-alone functions, assignment statements, and control statements. From this menu we choose the word SCALE. The system then presents a menu of the SCALE functions (SCALE_X,SCALE_Y or SCALE_XY). After we select SCALE_Y, the system asks which segments are to be scaled. An arrow on the CRT screen indicating the position of the tablet pen is used to point to picture segments. The set of segments selected is brightened on the screen; segments selected in error may be removed from the set. In our example, the segment SPRING is selected.

The next step is to select the origin point for the scaling operation. First we must choose between a static screen point, or one of the point valued functions. Since we wish the origin of scaling to be the point where the TOP and the SPRING join, we select the function POINT. We are then requested to select the desired point and do so with the pen.

Finally, with the aid of a menu we specify a real value to control the scaling operation. The menu, in addition to including input device names and real valued functions, allows us to type in model variable names, local variable names, constants or arithmetic expressions.

When the specification of the statement is complete we are prompted to generate the next statement. We may instead return to the editor to inspect the source, run the ACTION program, or use the DRAW program.

5. Advantages of the Interactive Dialogue Approach

The interactive graphical dialogue approach grew from our work in computer animation and music. We have found that it is readily learned and, equally important, a readily accepted method of interaction with a computer. The technique has several advantages:

a) Most action statements refer to picture segments and/or picture points. While it is possible to remember the names of all the segments when writing the ACTION program, it is almost impossible to specify in writing a specific point in a hand drawn picture. Telling the system that one wants "this" segment or "that" point, by pointing with the tablet pen is both easier and more natural.

b) As the user is constantly choosing statements, functions, or parameters from a number of visible and valid choices, many sources of possible error are eliminated.

c) The user does not have to learn any language syntax.

Direct entry of code by typing text is always possible. This textual method is used for the language structures such as arithmetic expressions, declaration statements, and control statements, that cannot be expressed naturally using the interactive dialogue.

The system is designed to allow the user to quickly switch back and forth from using the interactive graphical technique to using the text input method.

6. Conclusion

The ACTION programming system provides an easy to use facility of adding illustrative graphics and user control to models. With the system, one can use graphics to illustrate any stored time series data, data from previous runs of models, or statistical data.

The system demonstrates that the separation of graphics and models into independent tasks coordinated by the ACTION program simplifies the graphics programming.

The ACTION system is an experiment in the use of interactive graphical dialogue, an attempt to allow the use of this dialogue technique when most convenient, but allowing the use of typed text when the dialogue becomes awkward. The graphic technique can be expanded. For example, the source residing in the system can currently be perused only in textual format. However, a selected line of the program could be illustrated by intensity highlighting of the segments it affects and by arrows pointing to referenced points. Queries concerning the resident source such as "which statements affect 'this' segment" should be allowed. There will undoubtedly be much refinement to the graphic dialogue as experience with the system grows.

Acknowledgment

We wish to thank Marceli Wein and Nestor Burtnyk for their inspirational ideas.

References

1. Evans, K.B., Tanner, P.P. and Wein, M. A fast dynamic graphics package. Proc. DECUS, 4(3):817-821; 1978.

2. DeFanti, T.A. The digital component of the circle graphics habitat. Proc. NCC, pp. 195-203; 1976.

3. Futrelle, R.P. and Barta, G. Towards the design of an intrinsically graphical language. SIGGRAPH-ACM Computer Graphics 12(3): 28-32; 1978.

4. Tanner, P.P. Action programming system users manual. National Research Council internal report; 1979.

5. Tanner, P.P. Dynamic display of data. Proc. 4th Man-Computer Comm. Conf., Ottawa, Ontario. pp.22-1 to 22-7; May 1975.

6. Tanner, P.P. ACTION - a graphics aid to interacting with models and simulations. NRC-ERB 920, Ottawa, 1979.