To the Graduate Council:

I am submitting herewith a thesis written by Garry W. Amann entitled "MAKER: Figure Creator for the RASCAL Animation System." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

J. Michael Moshell, Major Professor

We have read this thesis and recommend its acceptance:

Charles E. Hughes

David L. Matuszek

Accepted for the Council:

Vice Chancellor
Graduate Studies and Reseach

MAKER: FIGURE CREATOR FOR THE RASCAL ANIMATION SYSTEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Garry W. Amann

March 1981

DEDICATION

To Amy, who gives meaning to everything in my life, and to Patrick, who is the future.

ACKNOWLEDGEMENTS

I would like to thank Dr. Mike Moshell for giving me the opportunity to work with him in the development of RASCAL, and for the support he has provided me. Also, I want to thank Dr. Charles Hughes for the excellent examples of what could be done with UCSD PASCAL.

ABSTRACT

A high school computer science curriculum that appeals to an average student must compete with television and with video games. Since video games are generally computer generated, the creation of video games and cartoons can be a tremendous teaching tool. The students need to be able to easily animate figures in order to make a cartoon or game. The RASCAL animation system allows them to do this while they learn UCSD PASCAL using an Apple II micrcomputer. The figures in RASCAL require a specific data structure which is difficult to set up by hand. Therefore, the system requires a program that will build this data structure for figures. The program must allow the user to see and change the figure interactively, and without the user having any knowledge of data structures. This program must be simple to use, so that is does not scare relatively inexperienced computer users. This thesis is a description of the program called MAKER, which answers these requirements.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

Overview

MAKER is an interactive graphics program which allows its user to create new figures and modify existing ones for use in the RASCAL animation system. The user can change the size, orientation and position of an entire figure in real time on the graphics screen in front of her. She can combine two figures to make a new figure. The shape of a figure can be changed by manipulating one line at a time. A figure is a line drawing since solid figures are not supported by RASCAL to be movable.

When working on single lines, the user starts working on the last line in the figure. However, she can grab any line in the figure. The line being worked on can be stretched to anywhere on the screen. Its color can be changed. It can be removed. The user can add it to the figure and work on a new line following the one just added. It can be defined as the diagonal of a box, which MAKER will draw into the figure.

The user can fill the background with a color or ask for help at anytime. She can put up a grid to help with exact positioning. Several figures can be loaded and displayed, and she can turn off figures as desired. Asking for a directory gives the names of all the figures in memory, showing each figure for two seconds as its name appears. The first color of a figure can be changed. A

figure can be given a new name. In general, a user can do almost anything to make and shape a figure to her needs, doing it without knowledge of data structures. MAKER maintains relatively bulky data structures in order to handle figures, but when told to save a figure, MAKER puts on disk only the data necessary for RASCAL to use the figure for animation.

MAKER uses three data structures in order to control figures and individual lines in figures. Turning on and off, showing, and orienting of figures is done by the RASCAL animation system. All other functions are handled within MAKER. The data structures are linked lists in dynamic storage.

The RASCAL figure head is shared with MAKER and is composed a singlely linked list. It is actually composed of two linked lists, but one is a subset of the other. The RASCAL head of each figure contains an ordinal value to identify it. It provides only the information needed by RASCAL, although MAKER accesses and modifies it.

The second data structure is a head node for MAKER. It contains the name, the ordinal, and a pointer to the RASCAL head. This structure also contains a boolean telling if the figure has been saved or not.

The line segment structure is a doubly linked list, and is the body of a figure. RASCAL traverses this list in SHOW, but all manipulation of the list itself is done by

MAKER. This structure provides MAKER's ability to shape figures. This is the only list that can go down in size by a significant amount, so a freelist is maintained for line segments.

Method of Attack of the Problem

Work on MAKER was started before the RASCAL line drawing routines existed. The original UCSD/Apple TURTLEGRAPHICS provided the only way to begin development. First plans for MAKER were made around the verb MOVETO.MOVETO allowed the user to move the cursor in X and Y directions. When RASCAL routines became available, this metod was dropped and a more cumbersome radial motion was required by RASCAL. However, as RASCAL became finished, input from users of Version 1 of MAKER indicated that the type of cursor control available through MOVETO would be much easier for the user than the existing method. Therefore, the final version of MAKER is very similar to the early ideas of how it should work.

There are basically two levels in MAKER, one being the figure level where the user can have several figures on the screen and all changes are upon the entire figure. Paddles are used to set a figure's scale and orientation. A keypad around the letter K is used to position the figure. All procedures available involve entire figures. During the entire development of MAKER, the ideas about how to implement this level have changed very little.

The other level is the line level where there is only one figure on the screen, and changes are made to individual lines in the figure. The user moves a cursor around the screen with a line rubberbanding from it back to the previous line. She positions the line as desired, then adds it to the figure. Work then begins on the next line.

The method of cursor control has been the most frequently changed idea in MAKER. The original method had it being moved by the keypad around K and the working line being made by a MOVETO to the cursor. When the RASCAL routines became available, MOVETO looked very awkward to implement and use. The easiest method of cursor control was to actually control the rubberbanding line by changing its length and angle. This could be done either by paddle or by keys. While some people found it easy to use this, others found it difficult. Most expressed a desire to be able to control the cursor by the paddles in an X and Y form. A MOVETO was implemented, and Version 3 of MAKER used the paddles to control the cursor in a joystick fashion, with the paddle values being scaled down to give a change in X and Y. However, the paddles did not work well as a pseudo joystick since the position for zero change was difficult to find. Therefore, Version 4 had the paddles give absolute X and Y values to the cursor.

Each new version of MAKER attempted to simplify its use while giving the user more power. This was the principal reason for changes in MAKER once it became able to create and save figures. It is aimed at users who have very little computer experience and do not need a complicated tool to scare them.

Layout of the Thesis

This thesis will present some background into previous work on figure creators. It will then present a detailed description of MAKER, first providing a user's manual, then detailing the implementation, including its data structures. In conclusion, it will consider what was accomplished and what future work can be done to improve MAKER.

II. BACKGROUND

In 1978, Dr. J.M. Moshell, Dr. R.M. Aiken, and Dr. A.L. Davis made a proposal to the National Science Foundation to develop a high school computer science curriculum. Software did not exist at that time, and the proposal was not funded. During the following year, Dr. Moshell headed a group which developed FELIX, an animation system on the Imsai 8080 microcomputer. A group of graduate students worked to develop an interactive object creator to work with FELIX, but work was not completed. This program was to be called MAKER.

With the FELIX software, another grant proposal was submitted. Dr. C.E. Hughes replaced Dr. Davis as one of the submitters. On the strength of FELIX, funding was provided for the proposal in 1979. Soon after the funding was approved, Dr. Hughes purchased an Apple II microcomputer. This computer and the UCSD PASCAL language it uses brought a wealth of ideas of that could be applied to a high school computer science curriculum. Therefore, it was decided to use the Apple for the curriculum. This meant starting over from scratch on the software, including MAKER. MAKER was an extensive well defined task, a good thesis for someone interested in it. It would involve helping to develop the animation system since animation would be necessary before MAKER could be finished.

The new MAKER would not be similar to the FELIX MAKER in that figures would be represented by line segments rather than individual pixels. An outstanding example of a computer animation system that uses line segments to represent figures was developed at the Chicago branch of The University of Illinois by Tom Difonti. It runs on a PDP-11 and is able to generate three dimensional figures. The graphics editor it contains allows the user to do anything she could desire to do to a figure. The RASCAL system works in two dimensions, and does not have near the speed of the ANIMA II system. However, a videotape of this system provided much of the inspiration for MAKER.

III. USERS MANUAL

MAKER is a graphics editor that is used to build figures for the RASCAL animation system. These figures are composed of lines. In MAKER, you stretch a line behind a 'cursor' that looks like a plus sign. When you have the line the way you want it, you add it to the figure. The entire figure can be turned, have its size changed, and be repositioned. All this is done by using the two paddles provided with the Apple II, and by single keystrokes.

At the bottom of the graphics screen there is a 'menu' displayed. It consists of words which indicate the 'procedures' you can use to build figures. These procedures are invoked by typing the first letter of the word being displayed. For example, one procedure called HELP is invoked by typing H. It switchs the screen to textmode and gives expanded definitions of the procedures available.

There are two basic menus in MAKER. The first one you see is used when working with entire figures. The other menu is invoked through a procedure in the first menu. It is used for working on individual lines in a figure. With a minimal amount of experience with MAKER, you can become very good at building figures.

Figure Control

Work is done on one figure (the work figure) at a time, even though you can have several figures in memory and on the screen and can switch back and forth among them to adjust each of them. The size and orientation of the work figure are set by the game paddles. Its position on the screen is set by a keypad. Below is a discussion of these controls.

PADDLE 0

If Button(0) is pushed, the orientation of the figure is set by the value of Paddle(0). The orientation of a figure is defined to be zero when the figure is first being created. After that, the angle of the figure is always in comparison to the original orientation. The point of definition of a figure is the start of the first line in the figure. All rotation of the figure occurs around this point.

PADDLE 1

If Button(1) is pushed, the scale of the figure is set by the value of Paddle(1). The scale can change the size of the figure from as small as a point to twice its size of creation. The size of creation is considered to be the size the figure is before you change its scale.

KEYPAD

The position of the figure on the screen can be controlled by a Keypad around the letter K. Typing the letter I causes the figure to move up one 'pixel'. Typing the letter M causes the figure to move down and to the left one pixel. Typing a period moves it down and to the right one pixel. The other letters in the keypad (U J , L O) cause similar movement in their direction from K.

ADDITION FACTOR

Moving the figure around the screen one pixel at a time can be very slow, so you can set an addition factor to help you move faster. Typing a number between 0 and 9 changes this factor. The figure is moved one plus the factor pixels every time a key in the Keypad is struck, so if you type a 4, the figure moves five pixels every time you type a key in the Keypad. To move only one pixel at a time you must type 0. When you start MAKER, the factor is set to zero.

Procedures

You make figures by invoking procedures with single keystrokes. The procedures available are listed in a menu at the bottom of the screen. The first menu you see in MAKER is shown in Figure 1.

```
G(ET S(AVE T(RNOFF B(KGRID W(RKON Q(UIT
D(IR F(ILL C(OMBIN R(ECOLR A(NEW  H(ELP
N(AME:
```

Figure 1. Main Menu for MAKER

GET

This is invoked when you want to get a figure from disk, or to get a figure that is already in memory as the working figure. After you type G, MAKER asks you for the name of the figure to be loaded. You should type the name in and hit the RETURN key. It then checks the figures in memory to see if there is one with that name. If there is not, then it tries to find a figure by that name on disk. If it finds the figure, MAKER loads it and makes it the current work figure. The previous work figure is left as it is, and cannot be changed until you GET it again. If it does not find it, there is no current work figure.

SAVE

When SAVE is invoked, it asks you for the name of the figure to be saved. It then finds that figure in memory and writes it out to disk. If there was a previous figure on disk with the same name, the new one replaces it. This does not change anything that you are working on.

TRNOFF

When you have several figures in memory, it can be beneficial to turn off some of them while working on the others. This procedure, TURNOFF, prevents the showing of the figure you tell it to turnoff. A figure can be turned back on by GETing it.

BKGRID

If you need reference lines to make a figure precisely, typing B draws a grid into the background of the screen. The lines occur every 20 pixels. Since it is in the background, figures move in front of it without erasing it.

DIR

Typing D will give you a directory of the figures currently in memory. It lists each figure's name and shows that figure for two seconds. This includes showing any figures that are turned off.

FILL

To get a new background color, you FILL the background. After typing F you are given a menu showing you the different colors and asking you to choose one. Because blue and black both start with B, black is referred to as DARK. FILL has an effect on the color of the grid that is drawn. The grid is drawn with white lines until a FILL white is done. Then the grid is drawn

with black lines until a FILL black is done. All other colors have no effect on the grid's drawing color.

COMBIN

This combines two figures by adding one to the end of another. It prompts you for the name of the first figure, then for the second figure. The combined figure retains the name of the first figure. MAKER adds a line of color NONE from the end of the first figure to the start of the second figure. The individual figures are no longer accessible. If you have a figure that you want to use several times, you should save it on disk, then get it off disk after each COMBIN.

RECOLR

RECOLOR changes the color of the work figure. It changes the color that starts the figure; but, if there is more than one color in the figure, it does not affect any color after the first color. For instance, if you make a figure with the first five lines blue, then use green lines for the rest of the figure, RECOLOR will only change the color of the first five lines.

NAME

To change the name of your work figure, type N. The figure is displayed and you are prompted for the new

name. All future references to the figure must be through the new name. The name of the work figure is displayed beside NAME:.

QUIT

QUIT allows you to leave MAKER. It gives you a listing of the figures in memory and prints whether they have been saved since they were the work figure. You then have the option of Returning to MAKER or Exiting it.

HELP

HELP changes the screen to text and gives a short explaination of each of the procedures available in the menu. When you have learned what you need, hitting the RETURN key ends the HELP procedure.

ANEW

ANEW starts a new figure by asking for the name of the new figure, then invoking WRKON on the new figure.

WRKON

This procedure performs all work on the individual lines of a figure. It can be invoked on the current work figure by typing W. It has its own menu, which is shown in Figure 2. Upon entering WRKON, all figures except the work figure are removed from the screen. A cursor that looks like a plus sign appears at the end of a line in the figure. This line is called the work

line, and all changes you make occur to it until that line is added to the figure. Its length and angle are controlled by the position of the cursor. When starting a new figure, the first line starts at the center of the screen. This start location can be changed by using the Keypad. The Keypad will move the entire figure (even if it is just one line) the same as it did when working on completed figures.

```
A(DD G(RAB R(EMOVE Q(UIT
B(OX F(ILL C(OLOR  H(ELP
```

Figure 2. Menu for WRKON

CURSOR CONTROL

When in the WRKON procedure, the game paddles cannot change the orientation or scale of the figure. Instead they are used to set the location of the cursor. PADDLE(0) controls its left-right position, PADDLE(1) controls its up-down position.

ADD

To build a figure, you ADD lines to it. You use Paddle(0) and Paddle(1) to position the cursor, and the work line stretches out to it. When you have the line positioned the where you want, and have it the proper color, you ADD it to the

figure. A line does not become part of the figure until it is added to it.

GRAB

This procedure allows you to get a new work line. The line you were working on disappears unless it has been added. GRAB prompts you with a menu asking if THIS-LINE is the one you want for your work line, or if you want to Quit GRAB. You should position the cursor over the line you wish to work on, then type T for This-line. MAKER finds a line that it thinks is the one desired, then starts changing the color of that line and asks if that is the line you wish to grab. If it is not, type N and MAKER continues to search for the proper line. If it is the proper line type Y. That line is now the work line. Its length and angle are determined by the position of the cursor. Since this is the work line, you must add it to keep it. You can add as many lines as you want in any part of the figure.

REMOVE

This REMOVEs the current work line and makes the line preceding it the current work line. This means the preceding line now stretches to the cursor from its starting point.

BOX

Typing B asks MAKER to draw a box using the current work line as a diagonal. What actually is added are four lines using the current color to form a box, then the work line is added using the NONE color. The next work line starts where the work line ended its diagonal.

FILL

FILL is the same as the FILL in the first menu.

COLOR

COLOR changes the color of the work line.

QUIT

This ends WRKON and returns you to the first menu.

HELP

HELP is similar to the HELP for the first menu, except that it contains information for this menu.

Quick Reference

Main Menu

PADDLE(0)	sets angle of work figure if Button(0)
PADDLE(1)	sets scale of work figure if Button(1)
KEYPAD	moves work figure 1 + Factor
GET	gets an existing figure for the work figure
SAVE	saves a figure on disk
TRNOFF	prevents the figure from being shown
BKGRID	draws a grid with lines 20 pixels apart
DIR	lists the names and shows all figures in memory
FILL	fills the background with the choosen color
COMBIN	adds a second figure to the end of the first one
RECOLR	changes the first color of the work figure
NAME	lets you rename the work figure
QUIT	ends MAKER
HELP	prints a page of information
ANEW	starts a new figure
WRKON	lets you change and add individual lines

WRKON Menu

PADDLE(0)	moves cursor up-down
PADDLE(1)	moves cursor left-right
KEYPAD	moves work figure 1 + Factor
ADD	adds the work line to the figure
GRAB	grabs an existing line for the work line
REMOVE	removes the current work line
BOX	uses the work line as a diagonal of a box
FILL	fills the background with a color
COLOR	sets the color for the work line
HELP	prints a page of information
QUIT	ends WRKON

IV. IMPLEMENTATION

Maker uses three data structures to implement its graphics editing. The one in Figure 3 is shared with RASCAL as a head node for a figure.

```
FIGHEAD = PACKED RECORD
            X : INTEGER;      (* X POSITION OF FIGURE *)
            Y : BYTE;         (* Y POSITION OF FIGURE *)
            OCT : BYTE;       (* ORIENTATION OF FIGURE *)
            ANG : BYTE;       (* ANGLE WITHIN OCTANT  *)
            ORD : BYTE;       (* POSITION IN ONLIST   *)
            INORDER : INTEGER; (* POSITION IN INLIST *)
            SF : INTEGER;     (* SCALE OF FIGURE      *)
            FIG : SEGPTR;     (* PTR TO START OF BODY *)
            FIGEND : SEGPTR;  (* PTR TO END OF BODY *)
            FPTR : FIGPTR;    (* PTR TO NEXT IN ONLIST *)
            INPTR : FIGPTR    (* PTR TO NEXT IN INLIST *)
          END;
```

Figure 3. RASCAL Head Node

The X and Y fields form a coordinate pair to give the location of the point of definition of the figure. The point of definition is the position where the first line of the figure starts. The OCT and ANG pair give the figures orientation in relation to the angle at which it was created. SF provides the scale of the figure. A figure is created at a scale of 100%, and can be scaled from 1% to 200% of its original size. The scale is represented internally with 256 being 100% and 511 being 200%.

The figure's position in the onlist is provided by the ORD field. The onlist is linked through the FPTR with lower ORDs being shown first. This makes them appear to be behind figures shown later. Only those figures that are turned on are linked into the onlist. The inlist keeps track of all figures that are in memory, and arranges them by their INORDER value. All references to figures are made by their INORDER value. The INPTR links this list together. The onlist is actually a subset of the inlist, but it does not have to follow the same order as the inlist. The two remaining fields, FIG and FIGEND, respectively point to the first and last bytes in the body of the figure. The RASCAL head node contains the amount of information needed by RASCAL to run properly, but MAKER requires more. It has its own head node, shown in Figure 4.

```
INLST = RECORD
          NAME : STR12;  (* DISK NAME OF FIGURE *)
          HEAD : FIGPTR; (* PTR TO RASCAL HEAD  *)
          HDNMBR : INTEGER;  (* SAME AS INORDER *)
          SAVED : BOOLEAN;  (* STATUS OF FIGURE *)
          FPTR : LISTPTR (* PTR FOR THIS LIST *)
        END;
```

Figure 4. MAKER Head Node

MAKER maintains this list primarily to have a disk file name for each figure. To know which name is attached to which figure, the HEAD field is required. Once a separate list is set up, it is easy to add more fields to ease figure manipulation. Since all RASCAL verbs use the INORDER value for identification, the field HDNMBR became obvious in order to save tracing through lists to find the value. The SAVED field is set to true when a figure is put on disk, and set to false when the figure is loaded by procedure GET or just begun.

The MAKER node list is traversed when MAKER is looking for a figure, causing minimal use of the RASCAL head node. When the node is found for the work figure, two global variables are set to point to each head node. Then, using these variables, changes are made to the X, Y, OCT, ANG, and SF fields of the RASCAL node, and the NAME field of the MAKER node. These changes are straightforward and occur when the user handles the figure. The only unusual procedure done to these nodes is when the user combines two figures.

Basically, combining two figures involves cutting the body of one figure and appending it to the body of another figure, as shown in Figure 5. In the following figures, cut links are shown by ⫽, and added links are shown by — — —. Minor changes must be made to the figure body in order to accomplish this properly. A line segment of color NONE is

added between the two figures to join them in their relative positions. Then the head of the second is severed. To prevent changes in the second figure resulting from differences in figure orientation and scale, the combined figure is given a scale of 100% and an angle of zero. The bodies are then parsed to adjust them so they will reflect this change.

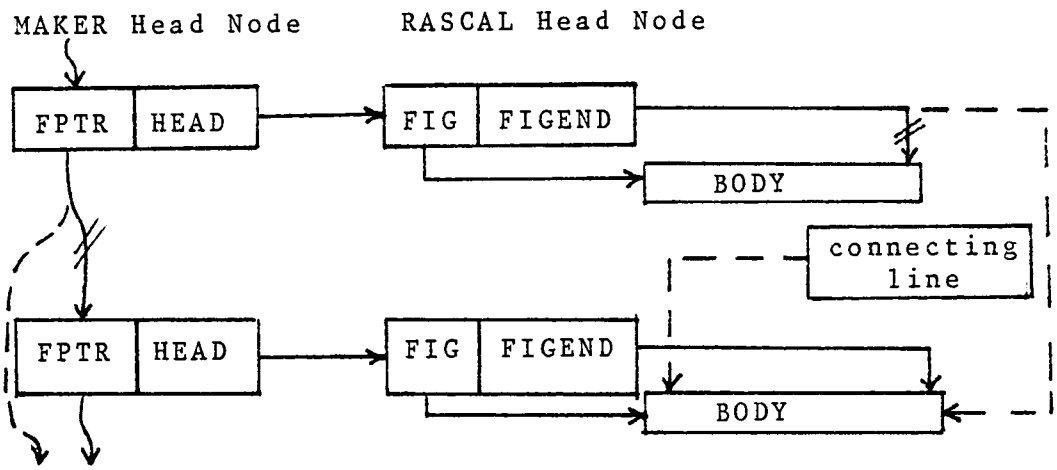

Figure 5. Combining Two Figures

The body of a figure is composed of line segments stored in the form shown in Figure 6. This form gives MAKER the ability to manipulate figures very quickly. The body is a doubly linked list giving great ease in adding or removing line segments. The five bytes, OCT through LR, make the essential RASCAL line segment. RASCAL uses OCT and ANG to set up its direction of movement, then moves the distance

contained in LR. If the line is rotated, LS is multiplied times the cosine of the new angle to get LR. LEN is referenced when scale is changed, being multiplied by the new scale to give LS.

```
LINESEG = PACKED RECORD
            FIL : BYTE;    (* SPACE FILLER *)
            CLR : BYTE;    (* OP CODE FOR LINE COLOR *)
            OCT : BYTE;    (* ABSOLUTE ANGLE OF LINE *)
            ANG : BYTE;    (*                        *)
            LEN : BYTE;    (* 'PURE' LENGTH OF LINE  *)
            LS  : BYTE;    (* LEN TIMES SF OF FIGURE *)
            LR  : BYTE;    (* LS * COS(LINE ANGLE)   *)
            JMP : BYTE;    (* HAS OP CODE 32 FOR JUMP *)
            FPTR : SEGPTR; (* ADDR OF NEXT LINE      *)
            BPTR : SEGPTR  (* ADDR OF PREVIOUS LINE *)
          END;
```

Figure 6. Line Segment Node

The CLR byte makes it possible to change the color of the line being worked on without affecting the other lines. When running RASCAL in real time, a color byte is needed only when the color changes. JMP holds the value 32, which RASCAL understands to mean to take the next two bytes as an address and jump to it. This works out conviently with having the forward pointer of the node immediately following JMP. The first byte, FIL, is a space holder to keep the number of bytes even. UCSD PASCAL will not pack records across word boundaries if the fields are of different types. And RASCAL cannot have any garbage bytes or it will get confused. So FIL is given the value of a color op code,

which is negated by the CLR byte. Table 1 has a listing of the current op codes used by RASCAL.

Table 1. RASCAL Op Codes

OP CODE	MEANING
0 - 7	Octant, pick up the next 5 bytes to generate a line
8 - 31	Color, change the drawing color
32	Jump, pick up the following two bytes as an address to jump to
255	Stop, do not go any farther

Line manipulation is done by working on one line at a time, and giving the user the ability to work on any line in the figure, adding and removing lines as desired. The user controls the absolute position of a cursor and the work line is stretches from the previous line to the cursor. The work line is referred to by a line segment called NOWSEG. When a line is added, the current NOWSEG is put into the figure and then a new NOWSEG is put in front of the previous NOWSEG.

Values for NOWSEG's OCT, ANG, LEN, LS, AND LR fields are determined in a procedure called RUBBER by using MOVETO. RUBBER first calls TRACE, which draws the figure up to the start of NOWSEG. Then a MOVETO is done to (CRSRX,CRSRY). The values for the fields are copied from a line segment that is used exclusively for MOVE, which is called by

MOVETO. The FPTR of NOWSEG points to CURSOR, the address of the start of the data that draws the cursor. The cursor's FPTR points to the line segment that follows NOWSEG. Figure 7 shows the linkage. The entire figure is then shown, including NOWSEG and the cursor. Showing involves clearing the screen, then drawing the figure. MOVETO is very slow in comparison to RASCAL line drawing, so TRACE does not clear the screen when it draws the first part of the figure. If it did, there would be an excessive amount of flicker due to MOVETO's slowness.

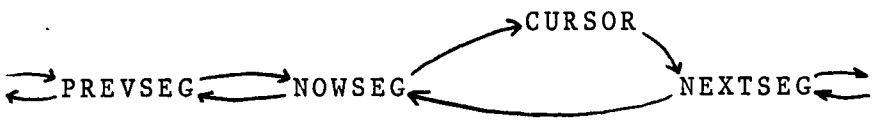

Figure 7. Linkage of NOWSEG

It is possible for MOVETO to call MOVE twice if the distance is greater than a certain length. If it does call it twice, it will only MOVE half the distance the first time. However, this presents a problem for NOWSEG. This is solved by having a line segment called TMPSEG around for just that purpose. TMPSEG's FPTR always points to the cursor and its BPTR to NOWSEG. All that need be done to use it is to change NOWSEG's FPTR to point to TMPSEG, then copy its data fields over. This linkage is shown in Figure 8.

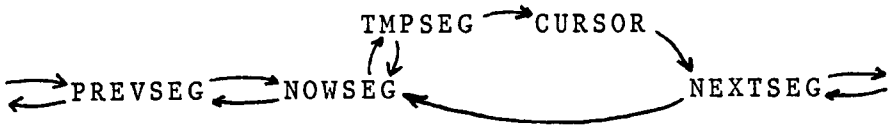

Figure 8. Linkage of TMPSEG

When a line is to be added to the figure, a boolean called ADDIT is set. RUBBER checks it after showing the figure. If true, it will invoke ADDSEG. ADDSEG simply changes the FPTR of NOWSEG (or TMPSEG if that is the case) to point to a new NOWSEG, and initializes the new NOWSEG. It also changes the BPTR of the line segment after the cursor to point to the new NOWSEG.

Grabbing a line involves unlinking NOWSEG and relinking it in the new location. Line removal is done by moving the links to NOWSEG back one line segment. This is shown in Figure 9. The only exception to this occurs when the first line is removed. In that case the links are moved forward one line segment.

Figure 9. Removing a Line Segment

One procedure available to the user is to define the diagonal of a box and let MAKER draw the box. This is implemented very simply using RUBBER. RUBBER does not check the cursor location, it just uses CRSRX and CRSRY. By setting them to the coordinates of the corners of the box, then setting ADDIT to true, RUBBER will add the line segments. On exiting from this level of work, NOWSEG and the cursor are unlinked from the figure. The global PREVSEG will hold the location of the line for NOWSEG to link in front of when this level is entered again.

V. CONCLUSION

MAKER is a tool which can display a figure being worked on, showing changes as they are made. The figure's position on the screen, size, and orientation can be adjusted. The actual shape of a figure can be changed by adding, removing, or otherwise adjusting individual lines. Figures can be combined to form a new figure. There are several useful utilities such as HELP, BKGRID, and FILL. All control of figures is simplified to the two game paddles and to single keystrokes.

The original goal of MAKER was for relatively inexperienced students to be able to create figures for animation. How well MAKER meets this goal remains to be tested, but feedback from current users implied that it has been reached. It is very easy to have features whose use is obvious to the designer, but not to the average user. This caused Version 1 to be too complex for most users. Each version after that resulted in a simplification of controls, until the necessary level was reached in Version 4.

Some ideas which were considered for MAKER but later omitted because of the complexity or awkwardness they involved. These included the ability to remove a line other than the current work line; movement of the cursor by relative adjustments, rather than absolute positioning; and selection of which input (keyboard or game paddles) controlled the cursor. The experienced user probably would

prefer to have a more complex version, since it is more powerful. Several features that have not been implemented but have been thought about are: grabbing a line at its center, then stretching it; being able to change the point of definition for rotation; and defining built in figures, such as a circle.

Several of the procedures in RASCAL were originally developed in MAKER when the need arose. Having been developed in conjunction with RASCAL, MAKER works very well with it. It is important that MAKER take advantage of any developments in RASCAL since these will probably increase MAKER's power while decreasing its complexity. Although MAKER is complete as it stands, it will continue to grow as ideas find a need to become reality.

LIST OF REFERENCES

LIST OF REFERENCES

Apple PASCAL Reference Manual; Apple Computer Incorporated, Cupertina, California; copyright 1979.

Apple Reference Manual; Apple Computer Incorporated, Cupertina, California; copyright 1979.

"Interactive Computer Graphics;" video cassette; Dr. Thomas Difonti; The University of Illinois, Chicago Circle; copyright 1978.

Users Manual and Report, second edition; Katheleen Jensen and Niklaus Wirth; Springer-Verlag, New York; copyright 1974.

VITA

Garry Wade Amann was born in Tampa, Florida on November 25, 1955. He lived in Dresden, Tennessee from his fourth grade year until graduating from Dresden High School in May 1973. He then entered the United States Air Force Academy in July 1973, and stayed there until August 1975. He then entered The University of Tennessee in Knoxville, and graduated from there in March 1978 with the Bachelor of Arts in an Economics major.

Mr. Amann returned to The University of Tennessee in January 1979 to begin work on the Master of Science in a Computer Science major. He will receive that degree upon the completion of this thesis, and will be going to work in Dallas, Texas as a Software Engineer for E-Systems, Incorporated.