Character animation with a computer

Jo Anna Timmerman

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
ROCHESTER INSTITUTE OF TECHNOLOGY

A Thesis Submitted to the Faculty of
The College of Fine and Applied Arts
in Candidacy for the Degree of

MASTER OF FINE ARTS

CHARACTER ANIMATION WITH A COMPUTER

by

Jo Anna Timmerman

September 23, 1989
Approvals:

Chief Advisor: James C. Ver Hague

Associate Advisor: Robert K. Keough

Associate Advisor: Howard Lester

Special Assistant to the Dean of Graduate Affairs: Philip Bornarth

Dean, College of Fine and Applied Arts: Robert H. Johnston

I, Jo Anna Timmerman, hereby grant permission to the Wallace Memorial Library of RIT, to replace my thesis in whole or in part. Any reproduction will not be for commercial use of profit.
# Table of Contents

ACKNOWLEDGEMENTS ................................................................. 1  
INTRODUCTION ............................................................................. 3  
CHAPTER 1 - Characters and Storyboard ........................................ 4  
CHAPTER 2 - Cel Animation and the Computer ................................. 7  
CHAPTER 3 - The Dubner CBG-2 .................................................. 9  
CHAPTER 4 - Starting Out - How to Draw the Characters .................. 15  
CHAPTER 5 - Animation on the Dubner CBG-2 ............................... 30  
CHAPTER 6 - Digitizing the Dream and Other Works of Art ............... 52  
CHAPTER 7 - Coloring the Characters and Adding the Backgrounds .... 57  
CHAPTER 8 - Names and Titles .................................................... 66  
CHAPTER 9 - Problems, Revisions and Lost Scenes ......................... 71  
CHAPTER 10 - Editing and Sound .................................................. 77  
CHAPTER 11 - Conclusion ............................................................. 80  
APPENDIX A - Glossary of Dubner CBG-2 Terms ......................... 81  
APPENDIX B - List of Dubner CBG-2 Keystrokes ............................ 85  
APPENDIX C - Printouts of KPL Programs ....................................... 93  
APPENDIX D - Bibliography .......................................................... 107
List of Illustrations

1a. The First Drawing ................................................................. 5
1b. The Final Characters .............................................................. 5
2. The Dubner CBG-2 ................................................................. 10
3. The Keyboard ........................................................................... 10
4. The Status Monitor Display of the CBG Subsystem ...................... 11
5. The Status Monitor Display of the Font Fixer Subsystem ............... 12
6. The Status Monitor Display of the Curve Drawer Subsystem .......... 13
7. RGB Display of a KPL in the Edit Mode ....................................... 14
8. A Set of Polygons Moved with <SHIFT-ROW> and Left Arrow Key ... 17
9. The Mouse and Tablet ............................................................. 21
10. The "Foreground" Pallete and "Background" Pallete in the Fixer Subsystem 21
11. The Pallete Display in the CBG Subsystem ................................ 23
12a. The Characters Before Cleaning in Fixer .................................. 25
12b. The Characters After Cleaning - Unwanted Lines Are Erased ....... 25
13. The Wife's Dress Pattern ......................................................... 26
14. The Wife Created as a Mask ...................................................... 27
15. The Wife with the Flower Pattern Added .................................... 27
16. The Sculpture ........................................................................... 29
17. The Status Monitor Display of the Animation Playback ............... 33
18. The Status Monitor Display of a Transform ............................... 35
19a. First Half of the Man's Walk Cycle ........................................ 38
19b. Second Half of the Man's Walk Cycle ...................................... 38
20. The Finished Texture Mapped Phonebook .................................. 44
21. The Opposite Page of the Phonebook Before Texture Mapping ...... 44
22. The Potion Book ................................................................. 46
23. The Toad .............................................................................. 47
24. The Princess ................................................................. 47
25. A Frame From the Dream Sequence ........................................ 54
26. The Final Sculpture of the Wife ............................................. 56
When I started this project, I had little idea of the amount of time it would take to complete. I counted up the time that I had worked on it and found that I had spent just under some 800 hours over the course of two years, not including the time I had spent writing the script. However, I could not have completed this film alone. Many people donated their time and experience to help me create this film and I would like to thank them all here.

First, I would like to thank the professors of my thesis committee, James Ver Hague, Robert Keough and Howard Lester, who advised and guided me through this project. They helped me to structure the film’s story and process and gave suggestions and encouragement to keep me going.

Next, I would like to thank the many people of PCI. To Chuck Munier, who edited the film together, gave many excellent suggestions for timing and pacing and was patient enough to endure a long session. Thanks also to Kristen Hartz and Eric McMaster, who assisted him in the video control room by keeping the tapes running. An extra thanks goes to Kristen, who carefully made all of my safety masters and dubs. And thank you to Todd Schafer, who edited both soundtracks with professionalism and fun.

Many thanks go to Ted Hummel, who allowed me to use the facilities at PCI to complete my thesis. I also want to thank Nina Widger, who taught me how to use the Dubner and gave me the idea to use the Magritte statue for the character of the sculpture from a short piece that she had animated. Thank you to Edward I. C. Kinney who was always there to laugh and who painted a few of the pictures himself. And many thanks to Mike Fien, who helped me out with technical problems and kept the Dubner running.
I would also like to thank Ruth Heitz and David Andrus for the enthusiasm they had creating the first soundtrack and for finishing it within such a short time.

A big thank you goes to John LaBarbera, who took the time from his busy schedule to create the second soundtrack. He created the perfect accompaniment which carried the theme and flavor that I wished to have for the film.

And a very special thank you goes to my husband, Erik Timmerman, who gave his support and love throughout the work of this film. He also guided me through the writing of the script and has taught me a great deal about animation.

All of these people played very important roles in the creation of this film. Without their support and encouragement, this film would not have been possible. I would like to give my humble thanks to them once more and hope that other students who plan such an endeavor in the future will be lucky enough to have such wonderful people around them to help out as I did.

Thank you all!
INTRODUCTION

My purpose in creating this work, "Of Myth and Magic", was to apply computer technology to the traditional cel animation process. During the course of my graduate career, I became employed at PCI, a local video production company, as a computer animator. I began training on the Dubner CBG-2, a three-dimensional animation system, creating logo and industrial animations for different companies. In general, these animations were designed to occur in three-dimensional space, using the Dubner's Curve Draw Subsytem. This Subsystem of the program allows the user to create and define objects with polygons and animate them with two methods - transforms or betweening. Transforms are a set of messages that instruct the computer how to move an object or set of objects. Betweening is an internal command that creates a series of transitional messages from a set of two or more recorded polygon messages. It was this method of betweening that most closely resembled traditional cel animation and thus I decided to use the Dubner to complete my project.

In an effort to expand the definition of computer animation, I chose an alternate aesthetic in which two-dimensional figures and flat color were used in place of three-dimensions, texture mapping or movement in space.

This report will explain the different methods I employed to achieve the effects and look of cel animation with the Dubner computer system.
Chapter 1
CHARACTERS AND STORYBOARD

The first step in creating this piece was to write the story and design the characters. I used standard dramatic structure in which the character takes action, he or his supporting cast have a reaction, he has a dilemma and then makes a desicion about what to do next. It seems that I myself went through the process of action-reaction-dilemma-desicion as I wrote this script. My storyline started out following one path of reasoning, but the farther down the line of the script, the less satisfied I was with what I had written. Thus the story changed many times - in fact, the only section of the script that was not changed from the original storyboard was the beginning, up to the point that the wife leaves the man.

The characters, however, did not change as much. I created the man first, and then patterned the wife after him. Both are loosely based on the Pink Panther's sidekick the Inspector. The dog started out as a scottish terrier, but after trying to draw him with the Dubner, I found that it would be difficult to draw a long-haired dog and also get effective emotional reactions from him. Many people have commented that he looks a lot like Snoopy, but I always remind them that Snoopy's ears are usually hanging down and that his legs are white. (See Fig. 1a and b)

The other characters in this film were created as needed in the film. They are described later on in this report.

While working on this film, I watched many cartoons. Like an apprentice painter, my masters in the field were Chuck Jones, Tex Avery and Hannah-Barbera from Warner Brothers and MGM cartoon fame. I must attribute some of the actions in my film to their genius. Tex Avery, famous for some of the most outrageous takes in character animation, inspired me to add bug-out eyes to the dog when he falls. Chuck Jones' cartoons of Wile E. Coyote helped me to name the magic shop, magic kit, toad and the list of companies in the phone book as ACME. And the writing in the ad and
Figure 1a
The First Drawing

Figure 1b
The Final Characters
potion book is in "Elmer Fuddese"¹ from the Elmer Fudd of Warner Brothers cartoons.

Chapter 2
CEL ANIMATION AND THE COMPUTER

Cel animation is created by drawing each frame of a motion onto a piece of acetate, called a cel, and then shooting each one onto film a frame at a time. There are several steps up to the finished product, however. The animator begins by first doing a pencil test of the action. After the preliminary sketches are finished, the animator films these drawings to check the flow of the movement and then can make any necessary changes. Once the animator is satisfied with the motion, he traces the line drawings to the cels. This process is called inking and is usually done on the front of the cel. The next step is painting in the color which is done on the back of the cel with a special paint called cel vinyl. The drawings are then checked for continuity and the backgrounds are also drawn and painted. Once the final cels are ready, the film is shot according to an exposure sheet which tells the camera man which order the cels are to be placed. After the film is processed, it is edited and the soundtrack is added.

I found using the Dubner CBG-2 Computer a viable alternative to traditional film animation and was able to translate the above process almost directly. It also afforded me some shortcuts. For instance, the Dubner plays back an animated sequence in real time, thus the time waiting for a pencil test film was eliminated. After checking a sequence, I could make changes almost instantaneously and try them again.

Usually, in a large production, many people are involved in each process. A key animator draws the extreme positions of the action and then another person draws the in-between frames. With the computer, I drew the extremes and the Dubner would create the in-betweens for me. It also drew each of the pictures, eliminating the need for the inking process.

Coloring cels is also a lengthy process. With the computer, a simple fill command is
used to color in the areas, and of course, there is no drying time. Also, in cel animation, several cels may be used to create a single frame. This results in a certain amount of "graying" of the image as the cels add a slight opacity. To compensate for this, the cels in the bottom layers must be painted more brightly than those at the top so that the colors appear the same. With the Dubner's two plane system, a character can be added to the background with no loss of color.

Also, with the ability to copy pictures or messages from one disk to another, editing the film was primarily accomplished during the creation of the animated sequences.
Chapter 3
THE DUBNER CBG-2

The Dubner CBG-2 is a very complex animation system. The program consists of three Subsystems: The CBG Subsystem, the Font Fixer Subsystem and the Curve Drawer Subsystem. With these Subsystems four different types of images can be created: text, pictures, points and animations. These can be recorded onto disk as separate and specific files or messages. There is also a programming language for the Dubner called Keystroke Programming Language which creates user-defined programs commonly known as KPLs. Additionally, there are internal commands called <DO> commands with which messages can be manipulated. In an effort to help the reader understand this system, I will describe the major points here and the details later as they relate to the specific tasks involved in creating “Of Myth and Magic”.

The Dubner hardware consists of a dedicated keyboard, a status monitor, two RGB color monitors, a tablet with mouse and the electronics rack which houses the microcomputer, memory boards, encoders and sync pulse generator. (See Fig. 2) The rack also holds two disk drives which take Iomega disks. These disks are flexible diskettes or “floppies” in a hard plastic case which can store up to 10 million bytes of information. In this particular setup, the memory boards hold 4 million bytes of random access memory (RAM) called Meg o’mem which can be used as a third disk drive. The program software can be stored onto any disk and the computer can be booted from any drive.

The user communicates with the program through the keyboard. The keyboard consists of a set of standard typewriter keys, arrow keys for the cursor, a number pad with an LED readout (referred to as the counter), and function keys which can perform specific instructions such as moving to a subsystem, reading or recording a message, drawing lines, duplicating or rotating a polygon, etc. (See Fig. 3) The status monitor reflects the keystrokes used and prompts the user when necessary.
Figure 2
The Dubner CBG-2

Figure 3
The Keyboard
The two RGB monitors display the images and can be used to show the palette of an image in one monitor and the complete image in the other. The tablet and mouse are used in the Font Fixer and Curve Draw Subsystems to control the cursor.

THE CBG SUBSYSTEM

This Subsystem is primarily used for creating and editing text messages. However, it can also display, manipulate and record pictures. In this mode, the user has control of two display planes, namely the foreground and background planes. This allows the user to read any combination of two pictures and/or text message at the same time. The artist can also add the elements of one picture or text message to another by a function key command called <WEAVE>.

Animation messages are recorded and played back in real time from this mode. Also, with the two display plane configuration, an animation can be played over or under a still picture. (See Fig. 4)

---

Figure 4
The Status Monitor Display of the CBG Subsystem
THE FONT FIXER SUBSYSTEM

This Subsystem deals with picture manipulation and font creation. The artist can draw with the cursor using the tablet and mouse. User-defined single or multi-colored brushes can be created and recorded in this mode. A picture can also be “cleaned up” by moving individual pixels or line segments with the cursor. (See Fig. 5)

![Figure 5](image)
The Status Monitor Display of the Font Fixer Subsystem

THE CURVE DRAWER SUBSYSTEM

*Points* and *polygons* are manipulated in this subsystem and are the descriptors for the objects the user animates. A polygon consists of two or more points to describe a line. A two point polygon will create a straight line, while a three or more point polygon creates a curved line. Polygons and points can be described in two or three dimensions. These polygons can be recorded into *sets* or groups, which results in the ability to affect all the polygons in a set at the same time. A set of closed polygons can be colored by using a *seed point*, a one point polygon, however, in the majority of this work, seed points were not used.

After the objects are created, they can then be manipulated by the use of function
keys for rotation, scaling, duplication, or weaving to the display plane to create a picture. The internal <DO> command MBETWEEN (multiple betweening), is used to create the incremental steps between one points message and another to create a new series of points. Transforms can also be used to describe the start and stop position of a points message to create animated sequences.

Though points messages are what the artist uses to create the animation sequences, they cannot be recorded into animation files. They must first be recorded as pictures. From a series of betweened points, an internal command called <EFFECT>A is used to read the points messages up and record them into picture messages. Transformed messages can be recorded either as points or pictures. (See Figure 6)

![Figure 6](error)

The Status Monitor Display of the Curve Drawer Subsystem

<DO> COMMANDS

These are internal commands that perform many useful functions such as copying messages, creating and playing animations, digitizing pictures, etc. They are invoked by typing the keystroke <DO>, and then typing a word or code word with specifications if necessary. These commands can be used in any of the Subsystems.
KEYSTROKE PROGRAMMING LANGUAGE (KPL Programs)

KPLs are user-defined programs in which the artist can describe any series of keystrokes and record that as a program. This language is very similar to BASIC in its structure, but also allows the user to invoke any <DO> command or function key that the program contains. The most beneficial use of this is in repetitive actions such as read two series of pictures up, weave them together and record them elsewhere on the disk. I found that a KPL could do anything I could do (except create the figures) faster and more efficiently than if I had tried to do it by hand. (See Figure 7)

At the end of this report, there is an appendix containing printouts of all the KPLs that are used in this project.

![Figure 7](image_url)

RGB Display of a KPL in the Edit Mode
Chapter 4
STARTTING OUT - HOW TO DRAW THE CHARACTERS

All the characters began as points messages created with the Curve Drawer Subsystem. A point is a location is space defined in standard Cartesian coordinates (x, y, and z) and a polygon is a collection of connected points which describe a line. A two point polygon creates a straight line and a three or more point polygon makes a curve. A group of polygons can be recorded into a set which allows the user to affect a number of polygons at the same time. As polygons are created, the computer keeps track of them in a list of polygons. This list is shown on the status monitor and shows the number of the polygon as it was created, its color, width, type and set number. A polygon can be any color within the palette and this is shown in the list by the palette position number. Width refers to the size of the line and is also shown by a number. For example a line width of two creates a very thin line, while a line width of ten draws a very thick line. The type indicates whether the polygon is describing a line or a one-point polygon called a seed point. A seed point defines the color of a closed area. The set number of a group of polygons is user-defined and does not have to be consecutive to the list, but is limited to 256 separate sets, numbered 0 to 255.

After designing the characters on paper, I started to draw them in points. This seems quite simple and straightforward, but to use these messages as beginnings for the extremes, I had to keep in mind the constrictions of the Dubner to make the between frames. The <DO> command, MBETWEEN requires that a set of extremes have the same number of polygons in the list and that each polygon has the same number of points and is the same type. Color, width and set number are irrelevant to this command, but if they are different from one extreme to another they will change at the beginning of each new extreme as they are MBETWEENed. Thus, to make a character walk across the screen, I had to draw both arms and both legs so that the computer would make the betweens properly even though in some pictures the back arm or leg would not show. This resulted in my method of creating all the relevant
body parts to a specific action for a set of extremes, betweening them and making the pictures and then erasing the unwanted lines later in the Font Fixer Subsystem.

There were other considerations to building the character. It was important to group the polygons of each body part into separate sets. By doing this, I could move or rotate each body part as a single object without affecting the rest of the character. Also, in view of final cleanup, the list order and color of the polygons had to be determined in advance of making the extremes. Since the computer will draw the polygons in the order of the list, I drew the parts of the character according to what was closest to the foreground. For instance, if the character was facing screen left, his right leg had to be drawn before his left one. Then, as the character walked from right to left, the lines of the left leg would pass across the right, thus “cutting” the unseen lines of the right leg. When these sets of legs are created in different colors, erasing a line became a simple fill command in the Font Fixer.

When the user first enters the Curve Drawer, he is presented with a cursor in the center of the RGB or display monitors and an empty list on the status monitor. The set number is defaulted to 1, the line width to 2 and the color number to whatever palette position was last logged out. The status monitor also reflects the x, y, z location of the cursor. To begin drawing a polygon, the user types the function key <INS-POLY>. On the display monitor, this results in a two point polygon with the first point at zero x, zero y, and zero z, and the second point ten pixels to the right which is selected or under the cursor. The status monitor shows that this is the first polygon in the list, its color, a width of 2, its type and that it is in set number 1. It also updates the position of the cursor to ten x, zero y and zero z.

Almost all of these parameters can be changed with function keys. By typing a number between 2 and 255 into the counter and typing the function key <RCD-MARK> the set number can be changed. Putting a number between 1 and 64 in the counter and typing <SELECT-CLR-#> results in the polygon changing color to that palette position. Line width can be changed by again placing a number in the counter and typing <BRUSH>. The type of the polygon cannot be changed as a polygon is either a line or a seed point depending on how it was created.

The list number cannot be changed unless other polygons are created and the first polygon is <DUPL>icated and deleted. The key <DUPL> always creates the twin of the selected polygon at the end of the list and numbers the copy of the set as 255. The original set is still existing at this point and the set number of the copy should be changed to a different set number, as consecutive <DUPL>icates will also place the new <DUPL>icated sets into set number 255.
To start a new set, a new number is placed in the counter and the keys <ALT> and <RCD-MARK> are used. If a polygon is selected with the cursor, that polygon is now part of the new set. If no polygon is selected, the next polygon created is placed in that set.

The key <DUPL> follows a very important concept called the Rule of Affected Polygons which controls the way sets are manipulated. This states that if a polygon within a set is selected all the polygons in that set are affected by whatever function key is pressed. If no polygon is selected all the polygons on the screen are affected. For example, if a polygon of a set is selected and the user types <DUPL>, the entire set of polygons is duplicated to the end of the list.

Another important pair of commands which follow the Rule of Affected Polygons are <ALT><DELETE-POINT> and <ALT><DELETE-POLY>. <ALT><DELETE-POINT> deletes the selected set, while <ALT><DELETE-POLY> deletes all the sets not selected. This can be a very important distinction if the user has not been recording points messages regularly!

Other commands that follow this rule are <SHIFT-ROW> with an arrow key, <CENTR-POINTS>, <REFLECT>, <ROTATE>, <SLOW-REVL>, <H(orizontal)-SIZE>, V(ertical)-SIZE>, and <BRUSH>. <SHIFT-ROW> with an arrow key (See Fig. 8) and <CENTR-POINTS> move sets of polygons around. <RELECT> turns sets

![Figure 8](image-url)

A Set of Polygons Moved with <SHIFT-ROW> and Left Arrow Key
around the vertical or y axis. <ROTATE> and <SLOW-REVL> turn the polygons counter-clockwise and clockwise (respectively) around the z axis when a number is placed in the counter. This number must be typed as one hundred times the amount desired. For example, if the user want to rotate a set by 90 degrees, 9000 would be typed into the counter. <H-SIZE> and <V-SIZE> also uses the counter to change the set's proportion, however these commands are limited to a ceiling of 200% and are typed in as such. The key <BRUSH> affects a set in the same way as described above for a single polygon.

The arrow keys move the cursor and its selected point around the screen. There are two methods in which the cursor can move: coarse or fine. In coarse mode, one hit on one arrow key results in moving the cursor 128 pixels. Each consecutive hit in the same direction also results in moving the cursor 128 pixels. However, reversing the direction moves the cursor one-half the distance of the previous move. Thus one arrow right moves the cursor 128 pixels in x, one arrow left moves it 64 pixels, a second arrow left moves it 32 and so on. In the fine mode, the cursor moves one pixel at a time, regardless of the direction. These two modes are toggled by pressing <SPACE-BAR>.

At this point, I have only described how to make a two point polygon or line. To create the curves, the function key <INS-POINT> is used. This key inserts a new point on a selected polygon after the selected point. Thus, if the cursor is on point number one of a two point polygon the new point is now number two and the original point number two is now point number three. Another line on the status monitor tells the user that the cursor is on point two of three which refers to the number of points in the selected polygon and which of the points the cursor is on. By using the key <CURSOR-TAB>, the cursor moves to the different points on the polygon.

More polygons can be added by using the keys <INS-POLY> and <INS-POINT>. Each polygon that is created is added to the end of the list with one exception. If in a list of polygons, the cursor is selected on a polygon in the middle of the list, <INS-POLY> will result in the new polygon being next in the list after the selected one. Thus if I had drawn the head and then the body of a character and had not drawn the back arm, I could go to the last polygon of the head and start inserting polygons to create the arm in the middle of the list.

I created profile views of my characters, beginning with the man. I drew his head and hat first as set 1, then the body as set 2. His moustache was drawn at the same time as his head, but knowing that sometimes he would look down, I <ALT><RCD-MARK>ed it into set 3 and <DUPL>icated it to the front. Then I deleted the original and changed the set of the new moustache from 255 to 3. Next I drew his front arm
in set 4. For his back arm, I selected the last polygon of the head and started drawing in set number 5. For his back leg and foot, I started set number 6 on the last polygon of the back arm. This leg was duplicated to create the front leg, moved to the correct position and changed to set number 7. However, this resulted in the front leg last in the list and in front of the body. To remedy this I duplicated the body, deleted the original and changed the copy back to set number 2. His front arm and moustache also had to be duplicated in the same manner. Thus, I had created the man with 7 sets in this order in the list: first head and hat as set number 1, back arm as set 5, back leg as set 6, front leg set 7, body set 2, front arm set 4 and moustache set 3.

I created a specific palette to work with to simulate pencil drawings and also to make it easy on my eyes. I chose a light blue for the background and dark greys and black for the lines. However, all the different sets were described with different palette positions even though the color might be the same.

I also determined the brush width at this point. I chose a width of 3 for all medium and long shots for the man’s body and a width of 7 for his moustache. Using a 7 brush eliminated the need for many polygons for his moustache and gave it a coarse feel. In close-up shots of the man, I changed the widths to 4 and 10 respectively. I felt that these brush widths were the proper proportions to the simplicity of the way I drew the character.

The wife and dog were created in much the same way. I started the wife’s head from the man’s by selecting the set of his head and using the <ALT><DEL-POLY> command. This left only the head and from that I added lips and eyelashes. Then, I deleted each polygon of the hat and inserted polygons to create her hair. From that point, I created the wife’s body, again placing each body part in a separate set.

The dog, of course, required a fresh screen and new polygons. I began with his head and body, tail, one ear and one leg. His other ear and legs required the use of duplication, new positioning and set and color changes to construct him properly.

With these three points messages, I had the beginnings of my main characters. The next step was to create the filled-in pictures and so that I would have a reference picture in color. Converting these points messages to pictures is accomplished by a simple keystroke called <WEAVE> and then recording the picture at a new message number. From there, I went to the Font Fixer Subsystem to clean up the unwanted lines and fill in the color.

The cursor in Font Fixer can be moved by either the mouse or the keyboard. I have
no preferences in moving the cursor in this Subsystem, so I will not designate which I used in explaining the methods of fixing pictures. However, the reader should understand the differences between using the mouse and using the keyboard. First, the mouse allows full freedom of movement in any direction according to where it is placed on the tablet. The arrows keys are singularly directional - one hit on an arrow key moves the cursor one pixel at a time. If the key <SHIFT-ROW> is used together with the right or left arrow key, the cursor will move 20 pixels to the right or left. If used with the up or down arrow, the cursor will move 10 pixels up or down.

The mouse has four buttons. The top button is for drawing and selecting colors from the palette, the right and left buttons are for zooming in and out respectively, and the bottom button is for calling up the palette to the screen. (See Fig. 9) In freehand drawing, pressing down on the top mouse button is preferable to using the arrow keys, but by positioning the cursor and using the key <SEG> a two pixel dot will appear on the screen. Or by using the key <WEAVE>, a user-defined brush can be deposited exactly once. Zooming via the keyboard is accomplished by the use of 6 keys named <COLOR BUTTON(>>. These keys zoom in by a power of two and increase the magnification up to 32 times normal size. The keyboard has two buttons to call up the palette <PAL-FGD> which shows the 64 position palette and <PAL- BGD> which gives the user 6 slider lines to change the red, green and blue, hue, saturation and value of a selected palette position. The mouse calls up the “foreground” or normal palette by one press of the bottom button and the “background” or slider palette by two presses. (See Fig. 10)

There are alternate methods of selecting colors besides using the palette on the screen. If the user knows the palette position needed, he can put that number in the counter and press <XFER>. This puts the selected palette position on the cursor. Another method is to put the cursor on a desired color on the screen and press the key <SELECT-BGD>.

Besides freehand painting, there are four main methods of cleaning pictures, all of which are keystrokes used in conjunction with the cursor. They are <FILL>, <ALL>, <DRAW-LINE> and <SHIFT-CHAR> with a right or left arrow. As mentioned above, the characters are created to isolate the parts of the body that will not be seen by drawing the foreground parts of the body over them. This creates a segmented line that can be <FILL>ed with the background color to erase it. By using <FILL> only the line that is selected by the cursor will be erased. This is more selective than <ALL>, which erases all lines with that palette position. Of course, it is not neccessary to use the background color - any line or area on the screen can be <FILL>ed or <ALL>ed with any palette position.
Figure 9
The Mouse and Tablet

Figure 10
The "Foreground" Palette (Top) and "Background" Palette (Bottom) in the Fixer Subsystem
<DRAW-LINE> uses two keystrokes to create a trapezoid which will connect two segments. This is useful to create a smooth line. The user positions the cursor over one segment of color and hits <DRAW-LINE> to mark the beginning of the trapezoid, then over a second segment. <DRAW-LINE> the second time thus creates the trapezoid in the color of the first segment.

The keystrokes <SHIFT-CHAR> with a right or left arrow is a fine-tuning method of cleaning. By placing the cursor on a segment of color and pressing <SHIFT-CHAR> with the right arrow, the segment length increases one pixel for each arrow key press. Conversely, using <SHIFT-CHAR> with the left arrow decreases a segment of color by one pixel.

Another important concept to consider at this time is palettes and palette positions. All pictures have a palette (which can be recorded as a separate message) and are “palette dependent”. (Points messages are not recorded with palettes.) By “palette dependent” I mean that whatever palette position the user chooses to draw with, the recorded picture will always show that area as being drawn with that position regardless of color. To put it another way, if a new palette message is read into the picture the position will remain the same even if the color is different. For example, let’s say I drew a circle in the middle of the screen with palette position number 3 which in palette message 50 is blue and recorded it as a picture. If I read in a new palette message to this picture and palette position number 3 was red, the circle would then be red.

The color of each palette position is determined or can be changed by the amount of red, green and blue levels specified. The range for each is 0 to 15, 0 being the least amount and 15 being the highest. Thus 0 red, 0 green, and 0 blue creates black, while 14 red, 12 green and 0 blue creates yellow. To change the values, the user puts a number between 0 and 15 and types R, G, or B on the typewriter key section of the keyboard. Each palette also has a status which can be opaque or transparent. This is determined by the numbers 0 or 3 respectively and the letter K is pressed. Usually, the only color used as a transparent color is the background or palette position 63, but any position can be designated as transparent. This is useful in weaving two pictures together such as the character and the background in the CBG Subsystem. The only other palette position that is predesignated by the Dubner is palette position number 64, which is the cursor color.

Colors of the palette can also be changed in either the Font Fixer or the CBG Subsystems. (See Fig. 11) However, pictures can only be woven together in the CBG which uses two display planes. This brings up one of the most important rules about palettes that most users (certainly myself) learn the hard way, which is to keep the
palettes of pictures to be woven together the same. When two pictures are woven and the palettes are the same, the resulting picture’s palette remains unchanged. But, when two pictures of different palettes are woven, the colors of the unselected plane are added to the palette of the selected plane in the resulting picture’s unused palette positions. For instance, say I took my circle picture in blue with its original palette, placed it in the background plane, selected the background and then put the same picture and put it in the foreground plane with the red palette. When I weave the two pictures together, the palette of the background plane is the same with one exception - the red color of the circle woven from the foreground plane is now in palette position number one. This seems harmless enough in this example, but if I have many pictures to weave together and they all have different palettes the problem increases dramatically and can get to the point that there are not enough positions to hold all the colors on the selected plane. Fortunately, in the CBG mode, the status monitor will flash a statement when two pictures are woven together telling the user if the weave was successful and whether the palettes were the same or were changed.

Keeping similar palettes of pictures to weave together is probably most important when it comes to changing them. If they are the same, a simple command is used to change them - when they are different, the user may have to go to each picture separately to change the color of an area within the series.
Thus, coloring the pictures of my characters is simple enough using these methods as described above. First I erased the unwanted lines by <FILL>ing them with the background color. Then I used palette position 61, a black color, and <ALL>ed the final lines. (See Fig. 12a and b) Next I <FILL>ed in the colors of each character. As I created the colors, I recorded a single, separate palette message and changed that message as I created new colors for each character and re-recorded the character pictures with the resulting palette. This way I could weave the pictures of my characters together and then change the colors again if necessary so that they would be aesthetically compatible. The original palette that I created in this stage eventually changed in the final piece, but by keeping the palette the same throughout this was a simple task.

The man and the dog were colored with flat colors, but I had different ideas for the wife. From a Warner Brothers cartoon called "People Are Bunny"\(^2\), I remembered a character of a game show host in which the character moved, but the plaid pattern in his suit remained stationary. Since the wife's dress covered a rather large area and thus in one color would look sort of plain, I decided to try to recreate this technique on the computer. The wife was also an appropriate character to try this out on as she was not in many scenes and so there would be no "overkill" of the idea.

Since the pattern had to be stationary no matter where the wife was, I knew that the flowers had to cover the screen. I began with a circular brush that I created with the <DO> command CIRCLE. This requires the user to indicate a radius in which I chose 15. This creates a circle in the center of the screen. I <FILL>ed it with a pink color and created a brush with the keystrokes <BOX> and <BRUSH>. <BOX> creates a moveable rectangle which surrounds the circle (or whatever the user chooses) and can be adjusted by selecting and moving the corners with the cursor. <BRUSH> then turns what is inside the box into a brush to paint with. Because the background is transparent and the circle is opaque, the brush is the size and shape of the circle. If there was an opaque color surrounding the circle, the brush would be rectangular. I made a flower using two colors - one for the center and another for the petals. Then I created a new brush out of the flower. I began the pattern by <WEAVE>ing the brush across the screen, moving the cursor with <SHIFT-ROW> and arrow keys. This created a regular pattern of one line of flowers. I could then make a brush from this line to cover the screen. Finally, I <FILL>ed the background with another opaque color. (See Fig. 13)

The next step was to set up the wife so that the flower pattern would be in her dress and not in the background. After cleaning up the unwanted lines and filling her colors

\(^2\) Ibid., pp. 269, 421. (Film: Warner Brothers, Merrie Melodies, People Are Bunny, release date: December 19, 1959, producer: Robert McKimson.)
Figure 12a
The Characters Before Cleaning in Fixer

Figure 12b
The Characters After Cleaning - Unwanted Lines Are Erased
in, including her dress, I went back to the CBG Subsytem, placing the picture of the wife in the foreground. I called up the foreground palette and selected the background color. By placing another palette position in the counter and pressing the key <SELECT-CLR-#>, I could reassign the background area to be that color, much like the <ALL> keystroke in Font Fixer. Then I selected the palette position of the dress and used the same keystroke to put the transparent background color in the area of the wife’s dress. This series of keystrokes creates a “mask”. (See Fig. 14) By putting the flower pattern picture in the background, the dress is now filled with the pattern. <WEAVE>ing the two pictures together adds the wife picture to the flower picture, at which point the opaque background area can be reassigned to the transparent background color. (See Fig. 15) This new picture is recorded elsewhere, saving the flower pattern separately.

My final main character is the sculpture. She is from a Magritte painting called “When the Time is Up”, from the book “Magritte” by Pamela Pritzker. She was created by digitizing or converting a video signal to bit-mapped picture. I first photocopied the color picture to increase the contrast and to make it black and white. The camera I used to digitize with is connected through the main editing suite at PCI, where I could preview the picture through the editing board. The Dubner program

---

3 Pamela Pritzker, Magritte, (New York: Leon Amiel Publisher, [no date]), plate #72.
Figure 14
The Wife Created as a Mask

Figure 15
The Wife with the Flower Pattern Added
has a special digitizing palette which contains a range of black to white values. To digitize, the <DO> command DIGITIZE (abbreviated DIG) is used. This also asks for a number of other specifications such as number of *bits*, which determines the number of grey levels in the picture and the area of the screen to capture as spelled out in *scanlines* and *dots*. There are 525 scanlines to the screen and 1024 dots or pixels to each scanline. The scanlines are numbered 1 to 525 from top to bottom and the dots 1 to 1024 left to right. I used 4 bits and designated only the area of the screen containing the sculpture. (The entire screen can be captured using less bits, however this results in fewer grey levels.) I decided to digitize the sculpture in four different sizes, knowing that some shots would require a close-up and others would need a long shot.

The next step was to clean up each picture and to separate it from the digitized background. I used a brush with the background color for the majority of the image, but her left side was nearly indistinguishable from the background. For this I went to the Curve Drawer to create a thick curved line to form her left side. I then saved this points message and <WEAVE>ed it to each digitized picture, changing the size when necessary. After recording these new messages, I separated each one into a series of pictures. These pictures contained only one palette position of the original cleaned and digitized picture. By doing this, I could fill in the “halftone” look into solid areas of the series of pictures thus also reducing the number of bytes each full picture would use. Then I used <WEAVE> in the CBG mode to recreate the pictures starting with the darkest value to the lightest. To finish the pictures, I read in the same palette from the other characters and changed the colors to a range of greys. (See Fig. 16)
Figure 16
The Sculpture
Chapter 5
ANIMATION ON THE DUBNER CBG-2

Since I had never created character animation on the Dubner up until this time, I decided to try a test run with the wife before leaping into the actual film. Thus her first on-screen appearance was standing with her arms crossed, frowning and tapping her foot. (At the time I thought this would be used in the film, but I never found a place in the script for it.) I began with the original wife points message in the Curve Drawer mode and changed the shape of her arms, deleting certain polygons that were her hands, and changed her mouth. I then recorded this new points message at another place on the disk as the first extreme. As a reference, I <WEAVE>ed this message to the background so that I would know the original position of her leg and foot. I selected the set of her leg and foot and rotated it 15 degrees clockwise, and moved it to match the heel of her shoe to the original position. Since this of course moved the lines of her leg away from her body, I selected the points of the tops of her leg and moved them back to match the woven picture. This second message or new extreme was recorded consecutively after the first. I then made more extremes from these two alternating messages, the last being the down position of the foot.

The next step was the DO command MBETWEEN. This was written as:

<DO>MBETWEEN,5040,5044,5100,1<ENTER>

The DO command MBETWEEN asks for first message (5040), last message (5044), where to record the resulting betweens (5100), and number of betweens (1). This resulted in a total of 9 points messages including 5 originals and 4 new betweens.

To convert these points into a series of pictures a command called <EFFECT>A is used. This is a series of keystrokes that reads up each points message, <WEAVE>s it to the background to make the pictures, and records it at a designated place on
the disk. It is desirable to have the correct palette read in at this time, as the pictures will be recorded with whatever palette is there. The keystrokes for this command are as follows:

1. Enter the first message number of the series of points in the counter (5100) and hit the key <XFER>.

2. Enter the first message number at which to record the pictures in the counter (5200) and hit the key <XFER>.

3. Enter the total number of points messages in the counter (9) and hit the keys <EFFECT> and A.

This results in the Dubner reading these points messages and converting them to pictures.

At this point, I could combine the pictures into animations. Since cleaning the pictures is a time-consuming process, all of the animated movements were checked immediately after the betweens and pictures were made so that if adjustments to the extremes were necessary, they could be done at that time. This is comparable to the traditional animator's pencil tests, but without the wait for the film to be developed.

To create the animations, the <DO> command COMBINE is used. This command asks for the first message number of the series of pictures (5200), the last message number (in this case 5208) and where to record the resulting animation pages (5300) and is written like this:

<DO>COMBINE,[OPTION],5200,5208,5300<ENTER>

There are several options with which the pages can be COMBINEd. They are:

- **H** Half Combine - which enables the user to play an animation in either the foreground plane or the background plane alone to view the cycle over or under a static image.

- **C** Coarse Combine - which duplicates every other scan line resulting in each picture having half the vertical resolution and enabling the animation to be played at a higher speed.

- **X** Semi-Course Combine - which is the same as coarse combine except that the first and last pictures are recorded in full resolution.
A  Alternate Combine - which takes each pair of pictures in the sequence and pairs the odd field from one picture with the even field of the next to make a single picture. This is the most efficient way to COMBINE an animation to play back at 60 fields per second and results in no loss of resolution. However, the animation must be played at speed 9, or it will jump up and down.

V  Very Coarse Combine - which duplicates very fourth scan line and reduces the resolution even more.

All of these options except for the half plane option are usually used for checking animations in which the byte size of the picture series is too large to play without a disk delay. A disk delay results in the animation “stuttering” as it plays, which means that the disk could not read the next page of the animation as fast as it could play the previous one.

To play the animation, the <DO> command ANIMATE is used and is written:

<DO>ANIMATE,[OPTION],5300<ENTER>

There are also a series of options which can be used with the ANIMATE command. They are:

F  Play the animation forwards (default)
B  Play the animation backwards (the message number in this case is the last animation page)
C  Cycle the animation continuously
H  Play the animation in one of the selected planes - this must be COMBINEd as a half plane animation
O  Oscillate or reverse the animation at either end
P  Pause the animation at the end of one cycle but remain in the animate mode
Q  Stop the animation after one cycle and return to the COMPOSE mode

These options can be used together or separately as desired.

When this <DO> command is used, the status monitor changes to the ANIMATION PLAYBACK DISPLAY. (See Fig. 17) This shows the name of the animation pages, the message number of the current page that is displayed or playing, the playback speed in frames per second, the running image count (or the number of times the
picture has been changed), and the elapsed time in minutes, seconds and frames. Also a prompt appears - "HIT COUNTER FOR ANIMATION". The counter digits 1 through 9 control the speed of the playback as follows:

<table>
<thead>
<tr>
<th>Counter Digit (Speed)</th>
<th>Images per second (NTSC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>7.5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>Stops the animation</td>
</tr>
</tbody>
</table>

These speeds can be changed while the animation is playing or can be incorporated into the animation in the COMBINE command. In this case, a speed message is recorded within sequence by hitting <RECORD>, a number in the counter and <ENTER>. This will result in the animation changing speeds automatically while in progress. Also, a pause message can be incorporated by hitting <RECORD>, zero,
another number in the counter for the pause (which is measured in fields) and <ENTER>. Thus a pause of 60 would hold the frame for one second.

To return the results of my first animation sequence, the playback was okay for a first try. However, it seemed too mechanical in that the foot did not rest but moved continuously. I went back to the pictures and added one more still of the wife with her foot in the original position or flat on the ground. This small change made the action playback more realistic. I also found that I could use the cycle option in the animation playback thus eliminating the need for so many pictures for this test.

The best speed for this playback was a speed of 5 or 6, but this was not, in my opinion, the optimal speed to create betweens for in the final film. Since video tape records at 30 frames per second, I felt that creating the betweens for a speed of 7 or 20 frames per second would be slightly smoother than the traditional film animator’s shooting on twos or photographing 12 frames per second. (Film is shot and projected at 24 frames per second.) Thus, I would be creating four less pictures than a film animator who shot on ones, gaining a smoother movement than one who shot on twos and not creating as much clean up and coloring work for myself that would be entailed in a playback of 30 frames per second.

With this test, I was ready to start the film. My first scene of the man with his hammer and chisel, however, became my first problem. When the Dubner creates betweens, it calculates the points to move in a straight line. Thus, when I tried to move the hammer arm, I found that no matter how many extremes I created, it would not move in perspective - the arm came out malformed or would move away from the body. I then tried a different approach, using the elbow as the pivot point and rotating the arm from there. This didn’t work either because the hammer wouldn’t reach the chisel. Finally, I tried moving the hammer and hand as a separate set and created the arm to follow. This worked, but then there was the problem of how the chisel arm should move. When I moved the chisel arm down in the extremes to match the position of the hammer arm, the betweens would move both arms at the same time. My only solution was to create the extremes as such, do the betweens and then go back to those betweens that weren’t moving properly and fix them by hand. Of course, this meant I had to be very careful not to accidentally delete those messages and thus I started the habit of saving all the points messages on one disk and the final pictures on another.

I also discovered an important restriction to rotating objects in extremes. Because of the straight-line betweening that the Dubner calculates, a set could only be rotated by 30 degrees in the z axis for the object to retain its shape. On occasion, I could push this with some objects up to 45 degrees, but this didn’t happen very often.
Objects such as arms or legs rotated to this degree often came out quite malformed in the betweens, so rotations over 30 degrees were reserved for faster moves and secondary objects.

To create the action of the man moving from one side to the other around the sculpture, I used another aspect of the Curve Drawer Subsystem - transforms. A transform message is a set of instructions that tells the computer how to manipulate a points or picture message. These instructions include origin of the object, size change, rotation, movement and eye location. A series of transforms designates key frame movements and can be applied to one message or a series of messages.

To define a transform, the user types <ALT>T, which brings the transform screen to the status monitor. (See Fig. 14) This screen is five rows with three letters each indicating the x, y, and z components of the instruction. The first row is the origin with the letters A, B, and C for the x, y, z co-ordinates, the second row is the percent size change with the letters D, E, F, the third is rotation with the letters G, H, I, the fourth is movement with the letters J, K, L, and the fifth is the eye location with the letters M, N, O. To change the values, the user types the letter associated with the co-ordinate. For example, to move an object in the x co-ordinate, the user types the letter J and then puts a number in the counter. The status monitor shows a prompt as to whether this movement should increment or decrement, at which time the user

![Figure 18](image)

The Status Monitor Display of a Transform
types <+> or <->. This places the value on the screen next to the letter. To invoke the transform once, the user reads in the points or picture message and types the letter T. By continuing to change the values and check their effects, the user can create the desired keyframe movements. These transforms can then be recorded onto disk.

Once a series of transforms is created, <EFFECT>T is used. This is an internal program which asks the user what message or messages are to be affected, how many resulting messages are to be created, where they should be recorded and in what form (points, pictures or more transforms), how many transforms are to be used and where they are. If there are only two transforms, the program will also ask if the change should be constant, or if accelerations or decelerations should be used, thus producing user-defined slow ins or slow outs. If more than two transforms are indicated, the computer will produce its own slow in and slow out.

In the case of the man’s motion around the sculpture, I created two sets of three transforms for each movement. I used the betweens that I have described above for the stationary position of his chiseling. Then I created a set of transitional extremes and betweens for his movement to “reset” his arm positions during the movement. In this way, I could reuse the original betweens for each position. To create this scene, I used <EFFECT>A to create the starting positon pictures. Then I transformed the series of transitional betweens with the first two transforms to make the next set of pictures. The third transform of each set was exactly the same as the second so that I could move the origina chiseling betweens to the new position. This sequence was repeated to move the man the second time. This method is quite similar to a film animator’s use of the pan bars to move a cycle of cels to different position.

From this point, I would like to describe to the reader the portions of the film that were the most difficult or ones that I used special methods in animating them. Please note that unless otherwise specified, all of the extremes were created in Curve Drawer and were betweened with the <DO> command MBETWEEN and the pictures were created with <EFFECT>A.

The next obstacle in this film was walk cycles. To assist me, I used the book “How To Animate Film Cartoons” by Preston Blair⁴, as it illustrated several walk cycles. This showed a keyframe breakdown of a one and a half cycle of a stationary walk to be used with panned backgrounds. In one-half of a walk cycle, Mr. Blair illustrates a character in five positions - contact, where the heel of one foot and the toe of the other foot are both touching the floor; recoil, where the first foot is on the floor and the other foot is just beginning to lift; lift, where the lifted foot is fully off the floor;

high, where the lifted foot is passing the other foot and the character has all of his weight on one foot; and a transitional stage in which the lifted foot is about to touch the floor again. The next step is again contact, only with the opposite feet. Also, the arms are moving in opposite directions of the feet; i.e. when the left foot is in front of the body, the left arm is in back.

I first worked on the man's walk. My first attempts on the half-cycle were okay, but my biggest problems were that the man appeared to stick to the floor or else slide his foot. I fixed this by matching toes and heels by $<$WEAVE$>$ing the previous finished extreme to the background to use as a reference for the next. I also found that since his feet were almost as long as his legs, his stride or contact position became very long in proportion to the lift and high positions. Thus when I used the same amount of betweens for all the extremes, he appeared to move very quickly in the stride and very slowly in the high position. I then tried several combinations of betweens, and came up with a pattern of 5 betweens for the contact to lift extremes, and 2 for the lift to contact. (See Fig. 15a and b. **NOTE:** These figures are separated slightly for clarification.)

Since I had the cycle of the right leg moving across the body, I could use these extremes to move the left leg. I moved each extreme of the cycle ahead with a single transform to put them in the correct position. Then I used $<$WEAVE$>$ to make pictures out of them as a reference. Next, I took each foot and leg and $<$DUPL$>$icated them. In this case, I did not delete the originals or place the copies in different sets as these were to be templates for each foot. Then I moved the left leg set as close as possible to the right leg set. By using the function key $<$NEAREST-POINT$>$ which moves the cursor and it selected point (if it has one) to whatever point is nearby, I could select and move each point in the polygon of the left leg to match the points in the right leg. I continued this procedure with the right leg, although in some cases, I moved the points or polygons by hand to eliminate a mechanical look. When I finished, I deleted the copies of the legs and then worked on the arms in the same manner.

Once this cycle was finished, I could then use it throughout the film, making minor adjustments in size, duration, or adding hats and other props as called for in each scene.

I decided to make the wife's walk much simpler. In contrast to her body size, I thought a tip-toe kind of walk would be funny, so her extreme positions only consisted of the contact and lift positions. However, her stride is much shorter than the man's and that created a big problem in getting the two of them on stage together. I tried to increase the number of betweens for her and decrease the
Figure 19a
First Half of the Man's Walk Cycle

Figure 19b
Second Half of the Man's Walk Cycle
between for him, but one or the other of them looked wrong. I decided to work with
the man’s correct number of extremes, and have them holding hands as they walked
onstage. To do this, I made a separate set of extremes of their outstretched arms,
maching the arms to the bodies as closely as possible. This worked out okay, but
my advisors commented later that the man still appeared to move too slowly. In the
final version, I removed some of the between of the man’s walk to speed him up,
and moved the wife’s between along with him with transforms. I also added his arm
to the wife’s extremes and deleted the arm from the man’s extremes. In this case,
the wife does appear to slide across the screen slightly, but is not as noticeable as
was the undesired effect of the man “walking through water”.

The dog walk cycle was created in much the same way as the man’s, taking into
consideration that the legs on each side of the dog had to move in opposite
directions.

Creating a finished animation sequence was not limited to using the Curve Drawer
Subsystem and COMBINEing the pictures. The other Subsystems were also
necessary to the process. For instance, once all of the walk-on cycles for the scene
of the man showing his work were created, I had to check timing of each character in
relation to the others. This required <WEAVE>ing the three series of pictures
together to create a new series. To do this, I used a KPL in the CBG mode. This
user-defined program places one picture in the background plane, and the other in
the foreground and uses the function key <WEAVE> to add the elements of one
plane to the elements of the other. In this case, the background is the selected
plane that tells the computer which plane to add the new information. Loops are
used in the KPL to control the number of pictures to <WEAVE> and record. By
creating this new set of pictures and COMBINEing them to make animation pages, I
could make the necessary adjustments to get everyone on the screen at the proper
time. This final series was saved on a separate disk as the final timings for the film,
while another disk held each of the separate elements.

Using cycles along with the <WEAVE> KPL also became a time-saving answer to a
lot of moves. For example, the dog wagging his tail was a matter of making one
points message of the dog body to use as a static, and three extremes for his tail.
This set of pictures could be used later in the scene by copying and <WEAVE>ing
them to the series of pictures of the man sitting up. By making another picture of the
dog without ears and a set of extremes and between of the dog’s ears moving, I
could make another series of dog pictures to <WEAVE> to the man petting him.

I would also like to point out to the reader that some eye movements like blinks or
looks in opposite directions were actually added to the pictures in the Font Fixer
mode. In my first attempts at this in the extremes, I found that a blink would look strange when it was between as the pupil would get progressively smaller. Similarly, an eye movement from one side to the other would result in the eyelid not always following the pupil. I felt that since most scenes were long shots, a simple change in the eyes from open to closed or left to right would suffice. I often used points messages to create these new positions, made them into pictures and finally brushes with which I could erase the original eyes and replace them with new ones.

Sometimes it was necessary to eliminate a frame or two to give a smooth feel to the action. For instance, when the wife hits the man, I created a set of 7 extremes with one between each for the wife and only 4 extremes for the man. When I made the animation for this sequence, I discovered that the action was too well-defined. Taking the picture out where the wife’s hand actually makes contact with the man’s nose made the action smoother and more realistic.

My first attempt at a three-dimensional move with these two dimensional characters is the dog moving his head from behind the sculpture. I used a picture message of the sculpture as a reference in positioning the dog. In this case, I only used the polygons of the dog’s head and body to create these extremes, as the rest of him would be hidden. I tried many different positions to get the dog’s head to turn, but none seemed to work; the problem being that his head would turn “inside-out” as the Dubner would between the points in a straight line. I then created a dog’s head from some scraps of paper, tape and a pencil to give myself a model to look at. I found that the head could turn in an arc with the right ear acting as a pivot point, instead of trying to move the head directly from one side to the other. This worked very well in the final betweens and started a new problem-solving technique. From that point on, whenever I could not visualize what kind of movements a character had to make, I would act it out, although since I was at work, I often shut the door!

I would like to discuss the dream sequence later in this report as this was a hand-drawn sequence that was digitized to create the final pictures instead of using extremes and betweens.

For the most part, two characters in the same scene were created as two separate series of pictures and then woven together. The scene of the man hugging the dog, however, called for special treatment. Because the man was holding the dog and the dog’s legs were hanging over the man’s arm, I could not easily make them into separate series of pictures and then weave them together. Thus, I had to make the extremes for both characters into one points message. First, I created each character with different sets for each body part as usual, but no similar sets between characters so that I could move all the parts independently. Then I posed the
characters together on the screen and made adjustments where necessary, such as moving the man’s head down far enough to his chest and extending his neck a little to make room for the dog’s head. The next step was to <WEAVE> this points message to the background as a reference picture.

Since I would be cleaning this series as a full picture and there were many overlapping parts, I had to determine which parts of the character were furthest back and what followed up to the foreground. The man’s arm was farthest back followed by the dog’s body and lower legs, the man’s body, his head, then the dog’s head, the man’s arm and finally the dog’s front legs. To create the final points, I recorded each layer as a separate message and erased the screen. Then I used the keys <ALT><READ-FGD> to put each message back onto the screen, starting with the man’s arm and the dog’s body. The keystrokes <ALT><READ-FGD> allow the user to add a new points message to the screen (and the end of the list) without affecting the points message that is already there. This was simpler than duplicating each set to the front as this was a pretty complicated scene as far as determining each movement in relationship to the other. Once complete, I used this message to create the rest of the extremes up to the point at which the man drops the dog. I then separated the two characters again into two different series of extremes.

Throughout this set of sequences, I used a silhouette of the sculpture as reference for the character’s actions and attention. (If I had used the actual digitized picture, the re-paint function of the screen would slow down considerably.) In the scene of the telephone book and the many other scenes that followed requiring props, I had to create the necessary objects for the characters’ actions. These props were created in points and then also recorded as a picture so that I could call them up whenever needed.

In the case of the man looking at the phone book, this prop also had to be part of the extremes like the dog as explained above. The hardest part in creating this sequence on the screen was figuring out what a person does with his other hand while flipping the pages. I finally acted the scene out myself and in finding the answer also got a better idea of how the pages should move.

The close-up scene of the phone book is one of the few sequences in which the foreground was not woven to the background. The book itself was created as a single picture and the motion of the hand was created separately to be combined as a half-plane animation to play over the book. The reason for this was the large amount of bytes that the book used as compared to the hand. If I were to weave the book and hand together, the resulting animation would probably have many disk delays while playing and also the pages would take up a lot of disk space. The
close-up of the potion book later in the film was created this way also, but these were the exceptions to the rule. Usually, the foreground action is too complex as far as byte size and would have disk delays if created as half-plane animations.

Creating the phone book picture itself was a fun and interesting challenge. I first created the text in the CBG mode. I used the Dubner’s Typewriter font for the main titles in the ad and Helvetica Bold for the body copy. The rest of the page listings were created with the Dubner’s smallest Helvetica Medium and the other ad was created with Futura and Helvetica Medium. After recording the text messages, I used &lt;WEAVE&gt; again to convert them into pictures. The boxes and lines were created with points in the Curve Drawer, and I put it all together with the Font Fixer.

The next step was to make the resulting flat page appear to have dimension and form with a process called texture mapping. This is done in the Curve Drawer mode. The 3-D shape is created in set of points called a patch which consists of four 4-point polygons. Then a box, which is a two-point polygon, is created in another set, which defines the area of the picture or texture to be mapped to the patch. To map the texture onto the 3-D points the keystrokes &lt;CTRL/M&gt; are used. This results in the picture shaped to the points, but also in erasing the points message, so the patches and boxes have to be recorded before invoking this command.

Patches are usually described in Bezier curves which are polygons with their points off the curve, instead of Spline curves which are polygons with points on the curve. To create Bezier curves, the user presses the function key &lt;ITALIC-&gt;. The keystroke &lt;ITALIC+&gt; will return the program to creating Splines. How the four polygons are created is essential to how the texture map will turn out. The first polygon in the list defines the top edge, and the fourth defines the bottom. The first point in all the polygons defines the left edge and the last point in all represent the right edge. The four interior points of the second and third polygons determine the amount of distortion of the image within this shape described by the other points.

The box points are created as Splines and describe the area of the texture diagonally from the upper left corner to the bottom right.

Patches generally should be created in the list after the box which defines the texture map area, but a patch does not have to have its own box. If there is no box immediately before the patch, the program will look for the next previous box and if none is specified, the whole screen will be mapped. This is a very important concept when connecting patches are used as I did for the phone book. By creating two or more sets of patches with boxes a larger 3-D area can be described and thus the user has more control over the object’s shape. These connecting patches must have...
common endpoints. The boxes also must have a common x or y coordinate with an adjacent box to result in a seamless texture map.

I used eight sets of connecting patches and boxes for the phone book. I tried to keep the size of the 3-D patches similar to the picture to be mapped as any increase in size would result in a lot of clean-up work. I moved the top and bottom edges of the book in x, y and z to make the overall shape and the middle points in the z axis to give it the same rounded look as a real phone book. This does require quite a bit of trial and error testing because the Bezier curves are somewhat deceiving as to how much they will affect the texture mapping. With each change, I also had to remember to re-record the points. Once I was satisfied with the results, I took the finished picture and created more 2-D polygons to outline the book. Texture mapping does create its problems, however. Since it the book was all text, I had to go back to the Font Fixer to clean it up and make it as legible as possible. (See Fig. 20)

The other side of the book was created in a similar manner with one exception. I created a full screen picture for the opposite page the same as the first in the CBG and Font Fixer modes. For this page, I used the Dubner fonts Lubalin Regular, Futura, Script and Helvetica Medium. (See Fig. 21) Then I reversed the picture with a Fixer Effect. This is a set of internal commands to manipulate pictures such as tilt and interp (scale), and a number of other commands to affect a font. The function key <EFFECT> is pressed along with the letter corresponding to the effect desired. <EFFECT> R reverses the image from left to right resulting in its mirror image. Then I went to the Curve Drawer mode and reversed the set of patches with the function key <REFLECT>. This turns the affected set around the y axis 180 degrees. I did not reverse the boxes though, as this would be defeating the process. The first point of each polygon in the patches still described the left side of the area to be mapped and by having the picture reversed resulted in it reading correctly when <CTRL/M> was used. I then moved this new picture to the left side of the screen to match the center crease in the book and wove them together.

The phone call itself again required the prop and the man in the same extremes messages like the man and dog hugging. Fortunately, from my experience of the man chiseling, I learned to move the phone first and then the hand and arm. This particular sequence took many extremes to get the phone up to his ear as the phone had to be rotated more than 90 degrees. However, when his call was finished, I could use the same extremes to put the phone back by copying them backwards and re-betweening.

The magic cloud that appears to change the dog into the magic kit in the next scene
Figure 20
The Finished Texture Mapped Phonebook

Figure 21
The Opposite Page of the Phonebook Before Texture Mapping
was created in points with extremes and betweens, but is added to the scene by a different method than <WEAVE>. When the cloud appears, the first few pictures replace the dog with the cloud so that the word “Poof!” can be read. The following pictures as the cloud seems to dissolve while it shrinks over the magic kit were added to the series with the <DO> command SECTION. This command, used in the CBG mode, isolates the intersections of two overlapping picture elements by assigning the intersecting area to a new palette position. The intersection’s palette position is the sum of the palette positions of the elements. Thus if a circle colored with palette position number six was SECTIONed over another circle colored with palette position number ten, the resulting intersection would be colored with palette position number sixteen. It is very important to make sure that the sectioned pieces are not going to equal a previously designated area, as usually these new intersections are used to show a “transparency” and the colors must be changed accordingly. SECTIONing affects only opaque or keyed colors; transparent or unkeyed colors like palette position 63 (the background color) are not added. The resulting image is combined on the selected plane as in <WEAVE> but retains the same palette as the background plane.

For all of the scenes in which the man reaches into the magic kit to pull something out I created the box as one points message and the man with the particular object as a separate set of extremes. Once the movement of him pulling the first object was finished, I could replace the object with a new one, making slight adjustments to his hand to fit. Of course, other elements and motions had to be added to each scene, but the basic motion was the same.

The potion book was created in the same manner as the phone book. The main titles were created in the Dubner’s Runnymede font and the body copy in Lario. (See Fig. 22) After texture mapping, however, this picture needed almost as much cleanup in Fixer as the phone book even though there was less copy. The problem in this case was that the fonts themselves were not that clean in the first place. Also, because I had given this book a slant in the x axis, the size of each line increased from top to bottom such that I had to clean them as if they were separate sizes.

At the time that I was creating the animations for this part of the story, I was also in the process of writing the scene about the magic dust. I knew that I needed a set-up for the magic curtain, so I created the two sequences of the man walking across the stage for the action with the dog to occur. This required an extra extreme to complete the cycle of the original walk, plus the addition of his hat and cape. When the first scene was finished, I could adjust the extremes to make the second. This entailed changing his smile to a frown by deleting the smiling moustache group and replacing it with a frowning one and then creating a cycle of his arm with a kicking
frog in his hand. These extremes replaced the orginal arm. I will discuss the action of the dog later along with the rest of the magic dust sequence.

The main characters were designed and created before I had started the animation. The toad and the princess, however, were created when their time in the story came. I wanted to make a truly ugly toad to warrant such a reaction from the man, but found this to be a difficult task. My husband, Erik Timmerman, suggested that the frog could have a pegleg and I added the wart, bad teeth and his lazy, though opportunistic, attitude. To create his smile in both shots, I made a static picture of his teeth to weave behind his moving mouth, making this foreground series into a matte, much like the wife's dress. The extremes of his arm moving to show the signs were reused in each scene except for the writing which was created in points to give them a "hand-scrawled" look. (See Fig. 23)

The development of the princess also came about when it was time to animate her. She did not come out exactly as I had imagined her originally as I hadn't done any previous sketches until I sat down at the Dubner. (See Fig. 24) However, I felt that she fit in with the other characters well, though she is much taller than the rest. It was quite difficult to manage the size of the stove and pot in comparison to her and the man. I decided to make her appear to stand on tiptoe to reach the pot. Her arm with the unhappy toad is another cycle used in both her standing position and her

![The Potion Book](image-url)
Figure 23
The Toad

Figure 24
The Princess
walk out. The man’s final reactions to her were created by setting up a few different cycles of him looking up and down and blinking and then weaving the sequences to the pictures of her walking out.

It may interest the reader at this point that it took a total of 87 hours to create the magic potion scene and the total running time was approximately 1 minute 6 seconds.

Now I would like to discuss the magic dust scene. Once I had finalized the script for this scene, I had to go back to fill the gaps in the magic potion scene. First, I created the sequence of the dog with the curtain box. I wanted the curtains to have a very realistic movement, so I moved each side of the curtains separately by hand. It would have been much simpler to move only one curtain, duplicate it as a set and then <REFLECT> the copied set to the other side, but that would result in the curtains appearing too much like they were drawn by a machine.

I would also like to point out to the reader that the girl from the man’s dream was created in points, traced from the digitized picture I had created. This was to establish her as part of the “reality” in the story.

My next problem came when I checked back the number of pages for the man’s walk which totaled 67 as compared to 111 pages for the sequence of the dog and curtain. Thus, in the final weaving, I had to make sure the timing of this sequence didn’t hold too long on the dog, which would slow down the story of the potion and yet still give enough time to establish the gag for the magic dust scene.

I added the dog to the kitchen scene next. This resulted in changing the original extremes of the man for this sequence as I wanted the dog to tug on his cape. I had to delete the cape points from the man and add it to the dog’s extremes. This worked out well, as the man is standing still from the time that the dog is in the room to get his attention. The dog’s contact with the man’s leg was created with a picture of the man as reference and then cleaned and colored properly in Fixer.

Up until this time, I had avoided trying to do three-dimensional turns in space with the Dubner except for the dog turning his head. Most of the other turns were created by switching the character’s direction from one page to the next and some were more acceptable than others. But I felt in this scene I should grit my teeth and try it. To turn the dog, I created a set of four extremes for one half-turn from right to left to a final position of facing the screen. Then I reversed all the extremes of this turn with <REFLECT> and recorded them elsewhere. The next step was to match the center extremes which I accomplished by weaving the last extreme of the first sequence to
the background, and changing this message to a new set number and a new color. Then I used <NEAREST-POINT> to move the points and polygons around so that the rest of the turn would between properly. After checking the animation, I found that it was best to delete the center position to make the movement smoother.

The scene of the dog pushing the magic curtain to the man was one of the more complicated sequences to figure out as it consisted of a dolly shot or a camera move in which the camera moves with the actor. A traditional film animator would create a stationary cycle of the character moving and then move the background with calibrated pan bars. My challenge then was to translate the pan bar movement to the computer. First I created a cycle of the dog pushing and a cycle of the curtain moving slightly. To emphasize the size and weight of the cabinet, I made many betweens for the dog’s cycle; a total of 170 pictures until his full stop.

My next problem was to create the moving background. I decided to use a chair as the visual cue, which also had to move in 170 pages, but I had to determine how far it should move per page. I also knew that the chair had to be big enough to remain in the scene while the dog was walking or the effect would be lost. My calculations then had to result in the total pixels for the chair to move from left to right and still remain within the title safe area.

A single step for the dog took 25 pictures. Then I divided the number of pictures for one step into the total number of pages (170), the result being 6.8 steps. Since the status monitor shows the user the x, y and z positions of the cursor, I was able to determine that the dog “gained” 145 pixels in one step. By multiplying the gain by the number of steps I found that the dog’s walk covers 986 total pixels. However, the total title safe area is 632 pixels. By subtracting this figure from the total pixels of the dog’s gain, I figured out that the chair should be 354 pixels wide. Then I created two transform messages that would move the chair a total of 986 pixels, adjusting the starting and stopping points so that it remained in the title safe area.

I went through this same series of calculations for the dog walking in front of the magic curtain to the man. I was very happy at the time that I could figure it out mathematically and the results were great. However, hindsight seems to be my worst enemy - at this writing, I think that I could have accomplished the same effect by making the start and stop positions of the background into extremes and betweening them to equal the number of pages for the dog’s move!

The dog’s zip out of the room was made by creating 4 points messages of broken circles for his legs and extremes for his neck to stretch out. After betweening the body and head, I used transforms with an acceleration on the move. I then tried to
transform the series of circle points to match the body move, but found that it wasn't an exact match. I had to move the resulting points series to each picture of the dog to get it to work.

The girl in this scene was fairly easy to work with as most of shots of her were close-ups or of her hands. I used the same technique to turn her around as I did the dog before, though she required less extremes. The sequence of her hand tapping the man on the shoulder was another cycle which was scaled with transforms to create the different sizes for each shot.

The sequence in which the man gets his hat pulled over his head was interesting as the more he was covered up by the hat, the less of his body points I had to use. By the time the man is completely covered by the hat, the only sets left of the original man were those of his hat, feet and a few polygons of his coat and cape. I did this with final cleanup in mind as the fewer polygons to erase the faster I could finish the scene.

The magic wand scene was the easiest to animate as the man was the primary character to move. I reused the points from previous animations to make him take the magic wand from the box. The rest of the scene was variations of the same motion of the man striking an object with the wand - a total of twelve in all, the last being his wife. I also reused the wife's extremes of hitting him, and adjusted his reaction extremes to add the hat and cape.

The scene of the wife stirring was created with the same props as the original kitchen scene, and included the same extremes for her getting angry and walking off as were used before. The scene with the man chiseling was also created in the same manner as the very first scene, though this time it went more smoothly.

The hardest part in this section of the story was the kiss - the problem being two characters with very large noses. Once I had them positioned properly, I had to design the extremes for the least amount of cleanup as I had in the scene where the man is hugging the dog. In this case, however, I did separate the two characters into two different sequences, keeping the man's right arm and left hand with the wife's extremes. Then I made another set of extremes of the man's eyes alone to use while he was kissing his wife and looking at the dog. This provided for a number of cycles and minimal cleanup.

The cycle of the dog's walk in was created by using the original extremes of the dog walking and replacing his head with one that included the magic wand. The final Curve Drawer animation for this project was for the credits. I reused all of
the walk cycles for the man, wife and dog (with the magic wand), decreasing them in size with transforms and adjusting them to look more like runs. In this case, since none of the characters had the same size stride, I centered all of the extremes for each with the function key <CNTR-POINTS>. This places the center of the affected polygons which is measured by the farthest x and y coordinates of the object at zero x and zero y. Then I had to move each in the y axis so that their feet lined up properly. Next, I added a seed point at the end of the list to create a silhouette from the outlines. A seed point is a one point polygon that fills the previously drawn polygons with a specified color.

After the extremes for each character were finished, I created another set to move in the other direction with <REFLECT> and betweened them to equal the same number of messages each. The movement across the stage was created with transforms at which point the pictures were made.

The resulting animation consisted of first each character runs across the stage separately, then the characters were woven together overlapping about half the series each, and finally overlapping so that they followed each other closely. This final sequence was created as a separate animation to cycle throughout the rest of the credits.

This concludes my discussion of animation by using the Curve Drawer to create the extremes and betweens of motion.
Chapter 6
DIGITIZING THE DREAM AND OTHER WORKS OF ART

As I have explained before, digitizing is the method of converting a video signal to a bit-mapped picture. Depending on the number of bits used, the resulting pictures have a softened, almost painterly quality. This, coupled with my investigation of traditional animation techniques as applied to the computer, directed me to trying my hand at digitized cell drawings for the dream sequence. I also took full advantage of the Dubner’s ability to dissolve between pages to enhance the dream-like quality of the scene.

My first task was to draw the cells. I started with tissue paper drawings, using an inking board with acme registration pegs. To create the first picture of the girl, I traced over the original photocopy of the sculpture and completed the rest of her body. Then I started to work on the main body of the sequence. I first drew the man turning his head while the girl looks on, drawing not only the motion, but also the stationary parts of both figures. After about a dozen drawings, I discovered that even though I was tracing the pictures, the bodies had changed considerably from page to page. Since I was going to re-trace the tissue drawings onto bond paper, I had to change this method to drawing only the moving parts of the scene. Thus, when I did the finals, I would trace the bodies from one drawing and the heads from the tissue series.

I drew the sequence of the man turning his head in much the same way that I had created the 3-D turns in Curve Drawer. I drew the pictures to the halfway point, then turned the tissues over to trace the general shape of his head for the second half of the turn. I also made some printouts from the Dubner of the man with his arm raised to trace so that the dream would look like the original scene with his wife. These needed some to be enlarged with a photocopier to match the size of my hand-drawn artwork.

The girl’s motions were fairly simple except to raise and clap her hands, and once
again I had to look into a mirror to understand the action. I drew only 2 pictures for each cycle of the girl clapping and kissing the man to use as a cycle.

For the bond paper drawings, I found a broad felt tip pen to coincide with the line weights I was using in the points messages. I used several, though, as they tended to mush down after a while. I taped the original pictures of the two characters to a light box and placed the tissue drawings over them to trace each cell. The final sequence consisted of 26 drawings.

Since I was going to use the digitizing camera in PCI's main edit suite, which was a flight of stairs above the Dubner room, I decided to use a KPL to digitize and record the pictures. Chuck Munier, who edited this film, gave me a series of commands to trigger the Dubner from the control board in the editing suite. A trigger is an option in the CBG mode (called <EFFECT>T) for the Dubner to recieve an external command to start an animation; generally used during a video tape edit. I created a KPL that used a loop to look for the external command with the keystrokes <WAIT>-1. After the program received the command, it would digitize the screen and record the picture. Thus, once I had found a digitizing option that I liked, I could control the Dubner from the edit suite and save myself from running up and down the stairs 26 times.

I used the inking board to register each frame once I had the picture centered and level. I chose 3 bits, or 3 grey levels, for the digitizing option. I found that this gave the lines an anti-aliased effect, and thus softened them. Normally, only 2 bits are used for a black and white picture to increase the contrast.

Once I finished digitizing, I colored the pictures in the Font Fixer mode. Normally, a digitized logo would need some clean-up work as the digitizing is not always exact. In this case, however, I chose to leave the lines in the pictures as they were and only cleaned up the backgrounds. The lines were made up of three palette positions, which I changed to purples and browns. Then I used the <FILL> command to color the characters. I chose pastels to emphasize the dream-like quality I wanted to portray. (See Fig. 25)

I added the original digitized sculpture picture to each frame in the CBG mode with a KPL to <WEAVE> a static to a series of pictures. I also stored all of the pictures in each stage - the digitized originals, the cleaned pictures, the colored pictures and the finals for the dissolve sequence.

A <DO> command is used to dissolve a series of pictures. It is written:

<DO> DISSOLVE, firstmsg, lastmsg, pause <ENTER>
The status monitor then prompts the user for a speed message in the counter to start the dissolve.

All of the parameters after the word dissolve are options, designated by numbers. If the user types <DO> DISSOLVE <ENTER>, followed by a speed, the computer will dissolve from the image on the selected plane to the message number in the counter. It will continue through subsequent messages until it finds a non-picture or non-text message, such as a KPL.

The first message option designates the beginning of the dissolve. The last message designates where to stop. The pause option is how long to stop between frames, measured in tenths of seconds. If there is no pause indicated, the computer will wait until a speed button is pressed.

For the main section of this scene, I chose to use a pause of 1 with a speed message of 5, as I was trying to show movement with the dissolves.

The pieces of art and sculpture in the magic wand scene were also photocopied and then digitized using the digitizing KPL. Probably, the hardest part about this was picking the artwork since I needed eleven different pieces. My sources were different
art books, including Frederick Hartt's "Art, Volumes I and II"\textsuperscript{5}, Gardner's "Art Through the Ages"\textsuperscript{6} and Edward Lucie-Smith's "Art Now"\textsuperscript{7}. My criteria in choosing were those that were most familiar and those that were outrageous.

Once I had the artwork digitized, I had to clean them. The new sculpture pieces were cleaned as the original sculpture by separating the darkest shadows from the highlights into separate messages and then putting them back together after cleaning. Some of the sculptures needed an extra shadow color to show more depth, which I added by hand. The Campbell's soup can, the Kabuki actor and the Arp piece required more careful clean-up work as the digitizing did not separate the colors.

I created a separate palette from the one of the overall film for each picture. This meant that in the final sequence, I had to keep changing palettes to weave them to the final sequence of the man. To compensate for this, I kept the palette positions of each sculpture in basically the same area so that I would not have to change the palette positions of the man in the scene.

The final digitized piece was the sculpture of the wife. I tried to create this sculpture by hand in the Font Fixer mode. I took the wife's body points message, added some polygons to finish the arm and neck and made a picture from the results. Then I made brushes from other digitized pictures and brushed them around the outline. However, when I compared this picture to the original sculpture I had to throw it out. My fake digitizing looked more like a fuzzy ball than the real thing.

Finally, using a printout of the outline picture, I traced and shaded the wife's sculpture with a pencil onto a piece of paper. I photocopied this to increase the contrast and digitized it. This came out fine, and with a little clean-up work became the final version of the wife's sculpture. (See Fig. 26)

Figure 26
The Final Sculpture of the Wife
Chapter 7
COLORING THE CHARACTERS AND ADDING THE BACKGROUNDS

Once the scenes were completed, checked as animations, and final revisions were made it was time to color the pictures. This was accomplished in the Font Fixer mode. In this chapter, I will point out some of the special problems and solutions I had in this process.

First of all, I started a new set of disks for the colored pictures. I copied the final messages or what I called the “outline pictures” onto a new disk a scene at a time. This way, I had a backup for my new colored pictures in case I made any mistakes.

The basic process was to eliminate the unwanted lines as described in the previous chapter entitled, “Starting Out - How To Draw The Characters” using the <FILL> and <ALL> commands. Since the palette used to create the outline extremes was mostly dark colors, I often changed those palette positions to different bright colors. This helped me to distinguish which lines were in front of others and thus which ones to erase. (See Fig. 27a and b) Once these were gone, I changed the final outlines in a series to palette position #61 which was black. Then I went back through the series to fill with color. This is similar to the film animator’s inking and painting only much faster.

To assist me, I created an interactive KPL which changed the cursor color at my command. With nested loops, this program kept track of the picture in the series, how many and which palette position to use next, and re-recorded the picture when finished. To change the palette within the program, I used the function key <XFER>. This places the palette position read into the counter onto the cursor. The program then “waits” for me to do whatever <FILL>s or <ALL>s I needed to do by a “trap” designated in the program by the key <INPUT-KEYB>. To continue the program, thus changing the color or re-recording the message and reading the next, the key <END-OF-MSG> is pressed. This program kept the number of keystrokes I had to
Figure 27a
The Original Palette with Dark Colors

Figure 27b
The Dark Colors Changed to Bright For Clarification
use to a minimum and made sure that I re-recorded the finals. Also, if I had a particularly intricate picture to clean up, I could re-record as necessary; or if I pressed <END-OF-MSG> too soon I could go back to the color through the palette or place the color number in the counter and use <XFER> myself. The reader should also be assured that once a pattern was established in coloring the pictures, I had few problems knowing which color was on the cursor.

When I came to a repeating cycle, I cleaned and colored the first series in the cycle. Then I went to the CBG mode. I placed the next outline picture of the cycle in to the foreground plane, and the matching colored picture in the background plane which was selected. This could be checked with the function key <BGD/FGD> which switches the pictures between planes. Once assured that the pictures were exactly the same, I put the number of the outline picture in the counter and pressed <DELETE>. This key always checks the user by placing a message on the status monitor “ARE YOU SURE?” and the user presses <DELETE> a second time to confirm or any other key to abort. Then I recorded the message in the background plane at the same message number. I could have created a KPL to perform this function, but I felt safer to do it by hand than to have the machine deleting my pictures. Also, since the cycles were usually short, the amount of time to figure out which ones to replace would have been the same as I still had to check each one.

Often a cycle needed to be moved, such as the walk cycles. To do this in the CBG mode, the user places a number in the counter and presses the key <SHIFT-CHAR> with an arrow key. This moves the entire picture in the selected plane the number of pixels in the counter in the direction of the arrow. Once I found the number of pixels to move the first picture in a cycle, I would move each of the consecutive pictures the same amount. This function was also created in a KPL to be used when I had to move a series of completed pictures to another place on the screen.

When two or more characters are on the screen at the same time and are not intertwined with each other, I colored each series separately and then used the <WEAVE> KPL to put them together. I kept each series on another disk to use later in the film if necessary.

The cycles of the wife were given special treatment because of the problems in matching a cycle to the pattern of her dress pattern. In this case, I cleaned and colored the first part of her walk cycles to the point of making each picture into a mask. Then I moved and recorded the pictures to complete the cycle and added the background still of the flowers to complete her dress with the <WEAVE> KPL. To eliminate the opaque background, I created another KPL for the CBG mode which performs the same function as the <ALL> command in the Font Fixer mode. This
program reads a picture, selects and displays the palette and puts the number of the palette position the user wishes to change into the counter. Then it positions the cursor at that palette number by using the keys <CTRL-C>. Next it reads the new palette position into the counter and presses <SELECT-CLR#>. This program can change as many colors as the user wishes and is valuable when dealing with objects that move across the screen. Using this program in the CBG mode eliminates the concern of where to place the cursor to <ALL> a series in the Font Fixer mode.

I created another pair of time saving KPLs to help speed up the process of cleaning. These programs used the points messages to create a new series of pictures of parts of the body that moved over many lines. By using this method, was able to color these moving sections quickly and spend less time erasing unwanted lines. For example, usually the hand and arm of the man moved over quite a few polygons. Sometimes in the extremes however, I had to change the set numbers of the hand from the arm in order to move them differently. Thus the first KPL changed the set of those polygons to be isolated to a set number different than the rest of the sets in the message. This KPL was applied to a copy of the original betweens as it re-records the points. I had to first figure out which polygons in the list were involved in the part by hand and recorded those on paper. Then I checked the set numbers to find a new set number to use for the hand and arm. Then I wrote the KPL. This is a more specific KPL than most as I wrote the list number of each polygon to be changed into the loop. The program selects each polygon by placing its list number into the counter and typing the keys <ALT> and <TAB>. Then it places the new set number into the counter and types the keys <ALT> and <RCD-MARK>. This changes the set number of the selected polygon and does not affect the rest in that set. The result of this program is a series of points messages with the hand and arm in a different set from the rest of the polygons.

The second KPL created the series of pictures of the isolated part. This program was more general than the first as it asked for a list number of one of the polygons in the set to be isolated. The program read the points message and selected the polygon with <ALT><TAB>. Next, it erased the rest of the polygons not in the set with the keystrokes <ALT><DELET-POLY>. Then it put a new number in the counter to record the resulting points as a picture message.

If the part to be isolated was in its own set and none of the other polygons in the message were part of that set, only the second program was used. The first program, however, could be modified to also make the hand a different color from the arm if necessary or to make two parts that did not intersect available to color at the same time.
One of the most effective examples of the use of these two programs was in the scene of the princess walking out with the toad. Her action is created from two cycles one of the toad squirming and one of her walk. With these KPLs, I created two series of pictures of these two cycles. First, I placed the polygons of her arm and the toad into one set. After making those pictures, I could modify the second program to delete the arm and toad set to make pictures of the rest of her body. This uses the keystrokes <ALT>DELETE-POINT>, which deletes the selected set. Then I could easily color and <WEAVE> the two cycles together.

Within a series of pictures, sometimes there would be a stationary figure or object. I created another KPL to add this object to a series of pictures as a brush. This program used the function key <WEAVE> in the Font Fixer mode.

Sometimes two brushes were used such as in each of the scenes where the man reaches into the magic kit. The first brush is the back of the box which was added before the man is cleaned. Then I cleaned a separate series of pictures of his hand reaching into the box and wove those pictures to the originals. After cleaning and coloring the rest of the man, I made another brush of the front of the box to <WEAVE> again.

Some effects in the film were hand drawn with the mouse in the Font Fixer mode. These included the scribbles above the man after he’s knocked out, the sprinkles from the magic wand and the zip-outs in the final scene. These were created by making a small square brush and drawing free-hand.

One very special hand-worked piece is when the bust of Constantine appears in the magic wand scene. I began with the cleaned digitized picture of the bust and made four new pictures - one of the head with no pupils, one of his pupils only, one of the head with the eyes cut out and one with the eyes closed. After cleaning the series of the man waving the magic wand, I wove the picture of Constantine with no pupils onto these messages. Then I made the pupils into a brush and added them to the bust according to the man’s motions so that Constantine appeared to be watching him. Then I used the picture of the bust with the cut out eyes as a brush to <WEAVE> over the series to complete the effect. The final picture of Constantine with his eyes closed was used when the man hits him with the magic wand. (See Fig. 28a and b)

Once all of the pictures were cleaned and colored, I created the backgrounds for each scene. Usually this required first returning to the Curve Drawer mode to create the props in points and then coloring them in the Font Fixer mode. In general, the outlines of these messages were drawn with a palette position different than the
Figure 28a
Constantine Watching the Man

Figure 28b
Constantine with His Eyes Closed
outlines of the characters so that if necessary they could be easily erased. These background pictures were saved as separate messages. The colors of the backgrounds were added to the palette of the final character pictures and from this, a new palette was recorded. I changed the series of messages to the resulting palette with the <DO> command PFORCE. This command is written:

<DO> PFORCE, palette msg, first msg, last msg <ENTER>

Then next step was to <WEAVE> the characters to the backgrounds in the CBG Subsystem. (See Fig. 29a, b and c) I modified the original <WEAVE> KPL to record the new message on the other drive. This is accomplished by reading the background still and foreground series from one drive, <WEAVE>ing them together and then selecting the other drive with the keystrokes <MSG-DRIVE-SEL> and the counter number <5> or <6> depending on the drive selected to record the new messages. The program then returns to the program drive to get the next set of messages.

One exception to this method of adding backgrounds was the scene where the dog is moving the magic curtain the man. After the character and curtain series was finished, I went to the original pictures of the outline of the moving chair. I cleaned and colored the chair as a still and then made it into a brush. Then I read each picture of the moving chair and wove the brush over its outline. After this series was complete, I wove it to the background of the floor and recorded this series onto the final pictures disk. Then I wove the series of the dog and the cabinet to the floor and chair and re-recorded the messages. This same method was used to complete the dog walking in front of the magic curtain to the man.

At the time that I was finalizing the pictures by coloring and adding the backgrounds, I had to start re-using my outlines disks. This required me to check and double-check that all of the pictures had been at least cleaned and colored before reformatting them to be used as final pictures. Fortunately, the colored pictures took up less space on the disks than the outlines, but this advantage was soon gone when I added the backgrounds and the digitized sculptures. At one point, I had a total of 37 disks filled with all the different messages, only four of which included the points messages. A total of 16 disks held the final messages which were used in the film.
Figure 29a
The Finished Characters

Figure 29b
The Background
Figure 29c
The Final Picture Composited in the CBG Subsystem
Chapter 8
NAMES AND TITLES

The characters of this film have no names. I tried to give them names, even ones that were just between me and myself, throughout the time I worked on the film, but to no avail. Thus, the reader may not be surprised to learn that it was almost as hard to name the film itself. I went through dozens of names such as “The Venus De Magic”, “The Artistic Alchemist”, and “Charmed, I’m Sure!” trying to find the right one. As the time grew short, it seemed that maybe I wouldn’t pick exactly the right name, but that eventually it would grow on me. And so, I chose the title “Of Myth and Magic”. This title is inspired from the title of the book “Of Mice and Magic”. But I felt that this was appropriate as many of the Warner Brothers cartoons occasionally made puns on other titles from other classics, including “Of Mice and Men” changed to “Of Fox and Hounds”, “A Star is Born” changed to “A Star is Hatched”, and “A Tale of Two Cities” to “A Tale of Two Kitties”.

I also wanted a series title like the “Merrie Melodies” or “Looney Tunes” from Warner Brothers and went through another set of titles such as “Algorythmic Tunes” and “Megabyte Magicals”. The title “Computer Capers” seemed to fit the best.

My own moniker, Joe Videoe, was the easiest. I had made it up in my first year as a graduate student when asked to do a business card on the Artronics Paintsystem. It is a pun on my own name Jo, as most people spell it with an “e” at the end. Since “video” also ends in an “o”, it seemed natural to add an “e” to both. I also made up a typeface then, based on the font “Babyteeth”.

To create this font on the Dubner, and to give it the ability to be easily rotated and moved, I used the Curve Drawer Subsystem. I made three sizes of squares and circles, the smallest being one-fourth the size of the largest. Then I constructed each letter from these shapes; each being easy to move as they were all in separate sets. The “v” was created as a triangle, the same size as the largest square. Once the
letters were finished, I made each into a separate set and moved them around to their final composition. (See Fig. 30)

I created a picture from the final points message and cleaned and colored them in the Font Fixer mode. (See Fig. 31) I created the edges and shadow using <EFFECT> E, which will create an edge and/or a shadow of whatever is on the screen. There are four different options for this effect:

- **E - Edge** - which will produce a rounded or squared off edge and also asks for the size of the edge, defined in dots and lines.

- **S - Shadow** - which asks for the number of iterations so that the user may create a single shadow or a stack of shadows to simulate 3-D. It then asks for the vertical and horizontal displacement and which direction the shadow should fall.

- **B - Both Edge and Shadow** - This option gives both, asking which should be on top. Then it asks for the same information as above.

- **T - Two-Color, Three-Dimensional Edge** - 3-D edges are specified like shadows with displacements and iterations. The result is a 3-D shadow that uses two colors to simulate a light source.

I made each of the letters into two separate series and edged one of the series. I put a three-dimensional shadow on the other series and then edged it also. The result was two messages of each letter, one with an edge and one with a shadow and an edge. I put the messages of each letter together with <WEAVE> in the CBG mode; the one with the edge only on top. Then I used <WEAVE> again to put all of the letters together to make the final picture.

To finish each, I created a trapezoid in points and made a picture from that. I used the Dubner font Handel to add the other lettering to the introductory title cards. (See Fig. 32a and b)

The main title frame was created in points also and cleaned and colored in the Font Fixer mode. I used the Dubner font Windsor for the words “Of Myth and Magic” and Helvetica Medium for the copyright notice. (See Fig. 33)
Figure 30
The Letters Created in the Curve Drawer Subsystem

Figure 31
The Letters Filled in the Font Fixer Subsystem
Figure 32a
The Final Title for "Joe Videoe"

Figure 32b
The Final Title for "Computer Capers"
Figure 33
The Final Title for "Of Myth and Magic"
Chapter 9
PROBLEMS, REVISIONS AND LOST SCENES

This film has been written and re-written many times. I had a complete story board when I began animating, but as I stated before, the only scene used from that was the beginning. Edward Kinney, an artist, good friend and co-worker, once told me “. . . to see what the painting will do to you.” I took this advice to heart; to the point that sometimes I even found myself frowning when the characters frowned and blinking in surprise when the characters were startled. There were times when I came up with new ideas at the Dubner that were funnier than those I had written. A finished portion of a scene would call for an even more outrageous outcome than I had thought of before; a movement that I thought I knew came out differently when I finished the animation. There were even scenes I created that didn’t work at all, but I still have them on disk. There is no full documentation of the final script, only piles of papers with notes scribbled on them and many of those not used. I would like to share some of those scenes and revisions with the reader in this chapter.

Often the problem of a scene was in timing or staging. I have received comments that the motion is sometimes too slow. This was best described in the scene where the man and wife walk onstage together, which I have explained in a previous chapter. I feel that the reason for the problem initially is that this was my first real effort in character animation. Also, I feel that if I had created the music before the animation, I would have had better success at pacing.

I also had scenes, however, that slowed down the whole story walk-outs that were unnecessary changed to dissolves between one scene and another. This was the case in the scene change from the man looking at the potion book to the scene of him in the kitchen. I originally had a medium shot of him walking off-stage screen right, but this was pointed out to me as slowing down the story so I took it out. The audience’s cue that he was going to move to another part of the house was indicated by him looking over his shoulder.
Another lost scene due to timing was when the man was talking on the phone. I had the man make gestures to describe what he was ordering. I used palette changes to dissolve up a magic wand, a top hat and then a rabbit in the hat. (See Fig. 34a and b) It was confusing, however, as to whether the man might actually be receiving these things as he spoke. It was also describing too much of the story - the audience knew from the phone book scene that he was ordering something magical and the next scene showed what he ordered. I cut this scene down to just the man listening and nodding.

From these experiences, I became a little more careful about a character’s reactions to something. For instance, I had some trouble in the scene of the man picking up the magic kit which has landed on the dog. At first, I started to second guess myself and thought about the man’s reaction to the flattened dog. The man’s relationship to the dog is meant to be a good one as the dog is his helper and friend. But, if the man were to stop and see what had happened to the dog, the story would stop also. The man could walk right on the dog, but I thought that would be too cruel for his character. I decided that the man would know enough not to step on the dog, but be too excited about getting the package to stop and worry about him at this point.

Some scenes were changed by simplifying a movement which would result in less pages to clean up. I had thought that the lid of the magic kit should be hinged, but once I started to make the extremes, I found that this would entail a lot of work. Instead, I decided to make the top a flat plane and have the man throw it away. Not only was this simpler to create, but it added a little extra humor to the scene.

The reaction of the man to the toad wanting a kiss was originally “overacted”. I had created four or five different faces for the man in this scene, some of them looking like he was more afraid than disgusted. In the final, I chose only one expression, which I felt had more impact than several.

In one scene, when the man throws the toad in the pot, I made the action too fast. From the walk across the stage, I had originally cut to the man walking to the stove and immediately throwing the toad in the pot. It was pointed out to me that the audience had no time to adjust to the new scene with this action. I changed this to the man walking into the kitchen, stopping and then throwing the toad in. This gives the audience a chance to “rest” and see what is going on.

Within this sequence of the man in the kitchen, I had some problems with staging. When the man walks in with the toad from left to right in a medium close-up, I first positioned the “camera” behind and a little to the right of the stove. When I went to the long shot, I positioned the man to be looking at the stove facing screen left. This
Figure 34a
Lost Scene  The Man Gestures a Wand

Figure 34b
Lost Scene  The Man Gestures a Hat and a Rabbit
is crossing the stage line, in which a camera would have to move from one side of
the set to the opposite. I had to change the position of the camera in the medium
shot to be in front of the stove, which could be accomplished by changing the angle
of the stove. I then also reversed the pictures in the long shot to fix the staging
problem.

When cleaning this set of pictures, however, I made a big mistake. I had not saved
the pictures of the man together with the stove in the medium shot, figuring I would
add it later. When I got to the point of cleaning the series, I had two different views of
the stove and unfortunately picked the wrong one. (See Fig. 35) I discovered this
after cleaning the both scenes of the man throwing the toad in the pot and looking
into the pot. When I looked at the next scene of the princess standing next to the
stove, it appeared that the stove had turned around. (See Fig. 36) Since the scene
of the man looking into the pot was a complicated one to clean because of the order
of his nose and the pot, I decided not to start over from the original outline pictures.
Instead, I made a single color brush of the stove, which was in front of him and
erased the stove from both scenes. Then I wove the color series over the outlines to
add the bottom of his coat that had been covered by the stove. I filled these areas in
with his coat color and then added the correctly positioned stove. (See Fig. 37)
Figure 36
The Next Scene of the Man and the Princess

Figure 37
The Corrected Scene
I have to admit, however, that after fixing this scene and realizing my mistakes in staging, I found that I did leave a scene in, which crosses the stage line. In the scene of the man talking on the phone, the “camera” is positioned behind the table. In the next scene, we are looking at him from the other side and he is facing the opposite direction. In the interest of finishing the film in good time, I left this in as I had no easy way to fix it without re-creating the entire sequence.

There were a few actions I had for the story that sounded like great ideas but ended up to be headaches to try to create. After the dog was dropped by the man in he scene after the dream, I thought the dog should stagger into the next scene. After several attempts, I gave up, figuring the dog had had enough time to “recover”. I thought it would be amusing if the dog slipped on a rug while pushing the magic curtain to the man, but realized this would be a lot of trouble for little gain and would slow down the story. I nearly had the man kick the dog out of the room in the magic dust scene, but found this to not only be difficult to do because of the size of his feet but also out of character for the man. I also wanted to have the man do a little soft-shoe step in the final magic wand sequence when he realizes that he’s hitting his wife on the nose. I tried three times to make the extremes, but wasn’t satisfied with any of them. All of these ideas had the best of intentions behind them, but given the limitations of time and sometimes patience, I felt I made the best choices to keep the story moving.
Once the scenes were finished, it was time to combine them into animations and edit the film together onto video tape. Because of the large byte size of each picture, combined animations of this film often would not play directly from disk without a disk delay. This meant that a single scene would have to be broken into many small animations that could be copied into Meg o’ mem and played from this drive. I was further disadvantaged in that I only had four disks left to hold the animations from sixteen picture disks. I would first combine each sequence onto Meg o’ mem, making sure that they would fit onto that drive as it usually holds only about 16 to 18 animation pages. Then I would copy the animations or recombine the sequence onto the animation disk that was available and note the message numbers used in the animation and the speed at which it should be played. I usually made breaks in the sequence at a hold whenever possible so that I could decide at the time of the edit how long to make the hold.

Most of the pictures disks, once combined into animations, could fit onto one animation disk each. Thus, during the edit I had to stop three other times to combine and ready the other twelve disks. Once I got started, this process took about an hour or so, as the number of pages that would fit seemed about the same each time. Plus, the counter reads off each page’s message number as it is combined, so that if I had chosen too many for an animation, I would know where to stop when I combined it again.

An average animation contained about 50 picture messages. The animation sequences lasted between 2 and 15 seconds each, and as the reader can imagine, the editing time involved was fairly long.

Chuck Munier, our chief editor at PCI, was kind enough to help out. He advised that we should edit the scenes together “hot” or live from the Dubner. Our alternative was
to put all of the animations to tape and the edit the pieces together onto another
tape, but this would result in a loss of a generation for the final master. He also
asked that I combine the animations with two picture messages overlapping be-
tween whenever I had to split up a continuous scene. In this way he could find the
proper matching point to edit the scene together.

The editing board that Chuck used was the Paltex “Esprit”. The Dubner was patched
in as the “source tape” and one inch tape was used for the master. Since the editing
board and the Dubner are in different rooms, the editing monitor was also patched
into my room so that I could see what was happening as I set up each animation.
We communicated by intercom.

The only editing “tricks” that were used in the film were dissolves and a circle wipe.
All the other editing commands were match cuts. For dissolves, I would place the
picture on the screen and Chuck would dissolve it up or down to the next frame. The
only exception to this was the dream sequence, where all of the dissolves were
controlled by the <DO> command DISSOLVE as mentioned before.

Chuck was also very helpful in editing the flow of the film. I asked him to give advice
on the lengths of holds and dissolves as I felt at that time I had been too close to the
film to be totally objective.

For the ending credits, I used the Dubner 20K Character Generator. I again used the
Dubner’s Handel font, which is compatible with the CBG-2. In this case, the anima-
tion of the characters running across the screen was one source, and the 20K titles
were the second. Chuck dissolved these titles up over the animation via the editing
board as the animation played. I reset the animation to play from the beginning for
each title.

We spent approximately 20 hours editing the film together, broken into one evening
and one day.

The next step was to have the soundtrack made. I wanted a player piano kind of
sound. I contacted Ruth Heitz who plays the piano with her partner, David Andrus
who plays the euphonium, in a musical team called “Duets Unlimited”. They said
they could create a soundtrack for me and would record it onto a cassette with which
I could edit. I provided them with a VHS copy with burned-in time code, so that they
could watch the film and take the timings. Ruth had an electronic keyboard linked to
a computer that she could compose with and then play back onto tape.

Ruth and David were able to turn the sound around within a weekend. They picked
some songs to go along with the scenes such as “It's a Long, Long Time” and “That Old Black Magic”. From these songs, and others, they were able to take a few bars and then improvise, so that there would be no problems with copyrights.

Next, I had to edit the music to the film. Todd Schafer helped me to do this. He recorded the cassette onto a one inch tape, so that he would have time code to assist him. The edit went well, except for some problems. Unfortunately, a cassette cannot hold as well to a specified time as it has a tendency to stretch as it plays. Thus, some of the music came out too short or too long for the visuals and had to be edited to fit. But Todd was able to make it work, and the film was complete in time for the show.

After the show, I decided to re-do the soundtrack. I called John LaBarbera, who has been in the Glen Miller band and also created the soundtrack for the original “Speedy” Alka-Seltzer commercials. After accepting the project, he asked for a three-quarter inch copy of the film, again with burned-in time code, as his system is also electronically controlled and hooked up with a three-quarter inch machine. John was able to provide me with a completely original soundtrack, using a calliope theme. When he was finished, he brought the track to PCI, where he and Todd Schafer edited it together using PCI’s 24 track audio editing board for final adjustments and sweetening. John also brought his computer and keyboard with him, so that I could ask for adjustments or changes if necessary. I made very few, as the track was excellent and we were able to finish the edit within a day.
I feel that this has been a very successful project. The Dubner CBG-2 is a very capable machine which enabled me to create the desired effects and look of cel animation. Working with a computer was much faster than trying to create the cels by hand. In particular, a considerable amount of time was saved due to the machine’s ability to create the in-between frames for me. And a lot of the guesswork was diminished as I could review an animation for changes immediately after I created it.

At the writing of this report, this piece has shown at the Animation Screening Room at the 1988 SIGGRAPH Conference and has placed second in the Student Division at the 1989 NAtional Computer Graphics Association’s Computer Animation Competition. Also, I have created three other character animation pieces for my clients at PCI.

If I had it to do over, I would make some changes in my approach to creating this piece. First, I would record the soundtrack and then animate the film. This would speed up the pacing, as I have had some comments that it is a little sluggish at times. Second, I would either re-write or completely eliminate the magic dust scene. I have shown it to many people, singularly and in groups, and have found that people are laughing at the other scenes more than this one. And third, I would have liked to have originally put it on film, which has higher resolution, rather than video tape, though this may have increased the amount of work that I had to do. However, I have gained a lot of experience from making this film and will apply this knowledge to future ones.
Appendix A
GLOSSARY OF DUBNER CBG-2 TERMS*

Bezier Curve - a type of polygon created in the Curve Drawer Subsystem in which the points are off the curve.

Bit - in digitizing, this refers to the number of grey levels that will be used in the picture. In more general terms, a bit is a binary digit, 1 or 0.

Box in the Font Fixer Subsystem, a box is a movable rectangle with which to capture an area on the screen as a brush. In the Curve Drawer, it refers to the polygon used to describe an area on the screen to be mapped. (See Texture Mapping)

Brush - in the Font Fixer, a user-defined area to paint with.

Byte - 8 bits. Each computer memory location stores one byte.

Counter - the four-digit numerical display on the keyboard.

Cursor - a marker on the screen that shows the user what he/she is pointing to, in regard to palette position or x, y, z co-ordinate.

Digitize - an option to convert an encoded video signal to CBG picture format.

Disk Delay - an effect of "stuttering" as an animation plays; the result of the disk being unable to read the consecutive animation page as fast as it played the previous one.

* This glossary was compiled from the CBG Operator's Manual, Program Version 6.2 as a reference to the reader for commonly used terms within the thesis report.
Display Plane (Foreground and Background) - a block of computer memory that shows one full frame or image. Background and foreground planes refer to the two image planes in the CBG mode, one displayed behind the other.

Dot - the smallest horizontal element of a scanline, also known as a pixel.

Field - one half of a frame. There is an “even” field (all even scan lines) and an odd field (all odd scan lines). Normal NTSC operates at 60 fields per second, pairing the odd and even field to create a single frame.

Font - a collection of characters in a particular type style and size that can be typed with the keyboard.

Hue - the shade or tint of a particular color or where the color “lies” in the spectrum.

Iomega Disk - a flexible diskette or “floppy” contained in a hard plastic case which can hold up to 10 million bytes of information.

List of Polygons - in the Curve Drawer, this list identifies the color, width, type and set number of the polygons as they are drawn. The polygons are also drawn in the order they were created in the list.

Meg o’ mem - in this Dubner CBG-2 setup, four million bytes of Random Access Memory (RAM), treated by the software as a third drive.

Message - any recorded CBG file pictures, text, fonts, points, KPLs, palettes, etc. - that is given a number on a disk.

NTSC - standard American television broadcasting format, as established by the National Television Standards Committee.

Opaque - an attribute of a color in the palette. Opaque colors cannot be seen through.

Palette - the table of 64 colors on a particular color display plane.

Patch - four 4-point polygons that describe a 3-D shape onto which a picture can be mapped. (See Texture Mapping).

Pixel - a “picture element” or a single horizontal unit on a scan line. (See Dot)
**Point** - basic element used in the Curve Drawer Subsystem; a location in space defined by standard Cartesian co-ordinates.

**Polygon** - a collection of points that describe a line or curve.

**RGB** - a form of “component” video, made up of separate red, green and blue signals.

**Rule of Affected Polygons** - a list of guidelines the Dubner uses to control which polygons are manipulated. Polygons are affected depending upon the selection of the polygon by the cursor, whether that polygon is part of a set, and if not a member of a set, if it is connected to other polygons. The rules state:

1. All polygons on the screen are affected if no polygon is selected.
2. All polygons with the same set number as the selected polygon are affected.
3. Only unnumbered polygons (those labeled “0”) connected to the selected unnumbered polygon are affected.

**Saturation** - the degree of purity of a color or the absence of white.

**Scan Line** - one of the 525 horizontal lines that make up a standard television picture.

**Seed Point** - a polygon consisting of one point. They define the color and type of fill of previously drawn polygons and are also used for sorting and hidden surface removal. “Quick Fill” seed points fill any area bounded by previously drawn curves and changes the color of the boundaries to the color of the seed point. “Normal Fill” seed points fill the curves outwards in all directions up to their boundaries.

**Segment** - one of the components of a CBG scan line. A segment has a length and a color indicating for how many dots that particular color runs.

**Selected Plane** - one of the display planes, either foreground or background, that is selected for the CBG to affect.

**Selected Point** - a point that the cursor is placed on in the Curve Drawer mode.
**Selected Polygon** - a polygon is “selected” when the cursor is on one of its points. When a point or polygon is “selected” it can then manipulated.

**Set** - when a polygon is inserted or created, it is assigned a “set” or group number. A set is used to manipulate several polygons at the same time. (See Rule of Affected Polygons)

**Spline Curve** - a type of polygon created in the Curve Drawer Subsystem in which the points are on the curve.

**Text** - a string of one or more characters.

**Texture** - a picture that is designated to be mapped to a 3-D patch. (See Texture Mapping)

**Texture Mapping** - a method in the Curve Drawer Subsystem by which a picture or “texture” can be shaped onto a 3-D points message called a patch. The area to be mapped is designated by a two-point polygon called a box.

**Title Safe Area** - a rectangular area covering approximately 80% of the usable area of a television screen, representing that part of the picture which (it can be safely assumed) will appear on everyone’s home TV set.

**Transform** - a set of instructions that tell the Dubner how to manipulate an object. Its controls consist of origin, size, rotation, movement and eye location.

**Transparent** - an attribute of a color in the palette. Transparent colors can be seen through if they are in the foreground display plane as if they weren’t there.

**Trigger** - an input from an external device (such as an editing board) allowing the device to start a real-time effect.

**Value** - the relative lightness or darkness of a color.

**Weave** - in the CBG mode, weave is a command which takes a two-plane image and turns it into a single-plane image. In the Curve Drawer, weave converts a points message into a picture message.
Appendix B
LIST OF DUBNER CBG-2 KEystrokes*

There are literally hundreds of keystrokes and combinations of keystrokes used in the Dubner CBG-2 computer. To explain them all, I would have to re-write the Dubner manual itself. Instead, I have compiled this list of pertinent keystrokes for each Subsystem referred to in the thesis report for the reader as a quick guide.

THE CBG SUBSYSTEM

<BGD/FGD> - reverse which of the display planes is on top.

<CTRL/C> - Set Cursor - in the CBG palette mode, move the palette cursor to the palette position number in the counter, 1-64.

<CURVE-DRAW> - enter the Curve Drawer Subsystem.

<DELETE> delete the message or font at the message number in the counter, on the currently selected disk. The Status Monitor prompts “HIT DELETE OR F”. Hit <DELETE> again to delete a message, hit “F” to delete a font. This keystroke is used in the same manner in all of the Subsystems.

<DO> - Enter the <DO> Command Mode.

<EFFECT>T - Trigger On - allow an external device to begin a real-time effect such as an animation.

<EFFECT>U - Trigger Off - real-time effects begin as soon as a speed is selected. (Default condition).

* This list of keystrokes was compiled from the CBG Operator’s Manual, Program Version 6.2 and the CBG Keystrokes Manual.
<ERASE-BGD> - erase the image from the background plane.

<ERASE-FGD> - erase the image from the foreground plane.

<FIXER> - enter the Font Fixer Subsystem.

<KPL-EDIT> - enter the KPL Editing Mode.

<MSG-DRIVE-SEL> - Select the message disk drive. This key is used in conjunction with a counter digit - <5> is the left-hand drive, <6> is the right-hand drive and <9> is Meg o' mem.

<ALT><MSG-DRIVE-SEL> - display the message drive directory.

<PAL-BGD> - display the palette of the background plane.

<PAL-FGD> - display the palette of the foreground plane.

<READ-BGD> - with a message number in the counter, read that message into the background plane.

<READ-FGD> - with a message number in the counter, read that message into the foreground plane.

<RECORD> - The first of two keystrokes to record a picture, text or palette message at the number in the counter. The Status Monitor prompts “RECORD WHAT?” The second keystroke can be any of the following:

<RECORD> record the current picture or text message in the selected plane.
<COUNTER 0> - record a pause.
<COUNTER 1-9> - record a speed message.
P - record the palette in the selected plane.

<ALT><RECORD> - re-record the text or picture on the selected plane at the same message number.

<SELECT-BGD> select the background plane.

<SELECT-CLR#> - in the CBG palette mode, this keystroke reassigns a palette position that the cursor is selected on to the palette position in the counter.
<SELECT-FGD> - select the foreground plane.

<WEAVE> - stamp the message in the foreground plane to the message in the background plane and put the result in the selected plane.

THE FONT FIXER SUBSYSTEM

<ALL> - change the color of all segments that are the same palette position as the segment selected by the cursor to the cursor or brush color.

<arrow keys> move the cursor in the direction of the arrow one dot or scan line depending on the arrow direction pressed.

<BOX> call up a movable box to capture an area of the screen as a brush.

<BRUSH> - turn the area on the screen designated by the box into a brush.

<CBG> - enter the CBG Subsystem.

<COLOR BUTTON(#) > - change the magnification. Magnification can be increased 2 to 32 times the original size by a factor of 2.

<CURVE-DRAW> - enter the Curve Drawer Subsystem.

<DO> - Enter the <DO> Command Mode.

<DRAW-LINE> - a two sequence keystroke, in which the first <DRAW-LINE> marks the segment under the cursor as one base of a trapezoid. When the cursor is moved to another segment and a second <DRAW-LINE> is pressed, the trapezoid is connected and its color is the same as the first cursor segment.

<EFFECT>E - Edge/Shadow - create an edge or shadow on the image in the display plane.

<EFFECT>R Reverse L-R reverse the image in the display plane from left to right.

<FILL> - change the color of the area under the cursor, including all like-colored segments directly connected, to the cursor color.
<KPL-EDIT> - enter the KPL Editing Mode.

<PAL-BGD> - enter and display the RGB-HSV or "slider" palette mode.

<PAL-FGD> enter and display the Font Fixer palette mode.

<SEG> - insert a 2-dot segment on the screen with the cursor color.

<SHIFT-CHAR> (with an arrow key) - with a right arrow, add a dot to the right side of the cursor segment, subtracting one from the segment on the right. With a left arrow, subtract a dot from the right side of the cursor segment, adding one to the segment on the right.

<SHIFT-ROW> (with an arrow key) with a right or left arrow, move the cursor 20 dots to the right or left. With an up or down arrow, move the cursor 10 scan lines up or down.

<SELECT-BGD> put the palette position number of the segment under the cursor onto the cursor to draw with.

<WEAVE> - deposit a user-defined brush exactly once on the screen.

<XFER> set the cursor color to the palette position number in the counter (1-64).

THE CURVE DRAWER SUBSYSTEM

<BRUSH> - define the width of the curve of the affected polygon(s) by the number in the counter.

<CBG> - enter the CBG Subsystem.

<CENTR-POINTS> center the affected points around the axis in the current view.

<CTRL/M> - invokes the texture mapping function.

<CURSR-TAB> - selects points in the selected polygon. Repeated <CURSR-TAB> selects consecutive points in the order of their insertion. When it selects the last point, it begins again. <CURSR-TAB> also selects a nearest point when no point is selected.
<ALT><CURSR-TAB> - select the point (in a selected polygon) whose number is in the counter.

<DELETE-POINT> - delete the selected point.

<ALT><DELETE-POINT> - delete all polygons belonging to the same set as the selected polygon.

<DELETE-POLY> - delete the selected polygon.

<ALT><DELETE-POLY> - delete all polygons EXCEPT those belonging to the same set as the selected polygon.

<DO> - Enter the <DO> Command Mode.

<DUPL> - duplicate the affected polygon(s) at exactly the same x, y, z location. The Curve Drawer assigns it to set number 255.

<EFFECT>A  Read Message, Record Picture - convert points messages to pictures.

<EFFECT>T - Key Frame Move - create an animation sequence with transforms.

<ALT>T - enter the Transform creation mode.

<ERASE-FGD> - erase a points message.

<ERASE-BGD> - erase a picture message.

<FIXER> enter the Font Fixer Subsystem.

<H-SIZE> - scale the affected polygon(s) horizontally to the number in the counter divided by 100. The range is 0 to 200.

<INS-POINT> - insert a single point into the selected polygon. The new point is selected for further manipulation.

<INS-POLY> - insert a two-point polygon at the cursor location and define straight lines.

<ITALIC +> - change the NEW POLY status to Spline.
<ITALIC -> - change the NEW POLY status to Bezier.

<KPL-EDIT> - enter the KPL Editing Mode.

<NEAREST-POINT> - move the cursor to the point nearest to it and select that point.

<READ-FGD> - read a points message.

<ALT><READ-FGD> - add the points message in the counter to the displayed points message. The new points are added to the end of the list.

<READ-BGD> - read a picture message.

<RCD-MARK> mark the affected polygon(s) with the set number in the counter.

<ALT><RCD-MARK> - assign a set number to only the selected polygon even if it is member of a set. The succeeding polygons will be inserted with the new set number. If no polygon is selected, <ALT><RCD-MARK> assigns a new set number to the default NEW POLY value.

<RECORD> - with the following keystrokes <RECORD> records a particular message:

<RECORD> records a picture message.
P records a points message.
T - records a transform message.

<REFLECT> - turn the affected polygon(s) about the vertical axis.

<ROTATE> - rotate the affected polygon(s) counter-clockwise by the amount placed in the counter divided by 100.

<SELECT-CLR-#> select the palette position in the counter and assign the values to the affected polygon(s).

<SHIFT-CHAR> (with an arrow key) move all the points that are at the same location as the selected point.

<SHIFT-ROW> (with an arrow key) - move the affected polygon(s) along the same x, y, z axes.
<SLOW-REVEAL> - rotate the affected polygon(s) clockwise by the amount placed in the counter divided by 100.

<SPACE-BAR> - toggle cursor movement between fine and coarse mode.

<TAB> - select the first point of a polygon. Repeated <TAB> selects consecutive polygons. When it selects the last polygon in the list, it begins again.

<ALT><TAB> - select the polygon whose list number is in the counter.

<V-SIZE> - scale the affected polygon(s) vertically to the number in the counter divided by 100. The range is 0 to 200.

<WEAVE> - convert the points image to a picture image.

<DO> COMMANDS
(Options for each <DO> command are designated by [ ]).

ANIMATE  replay an animation sequence.
  <DO> ANIMATE [.option] [.msg] <ENTER>

CIRCLE - make a circle or an ellipse of the specified size and color.
  <DO> CIRCLE [,lines [,dots [,palpos [,thick]]]] <ENTER>

COMBINE  combine a series of messages into a series of animation messages so that they can be played in the animate mode.
  <DO> COMBINE [.option], firstmsg, lastmsg, newmsg [,drive] <ENTER>

DELETE - delete the specified messages or fonts.
  <DO> DELETE [,font#], firstmsg [,lastmsg [,drive]] <ENTER>

DIGITIZE - convert an external video signal into a CBG picture.
  <DO> DIGITIZE [,bits] [,dot#, #] [,line#, #]<ENTER>

DISSOLVE - dissolve through a series of messages.
  <DO> DISSOLVE [,firstmsg [,lastmsg [,pause]]] <ENTER>

PFORCE - force a palette into a message or series of messages.
  <DO> PFORCE, palmsg# [,firstmsg [,lastmsg]] <ENTER>
MBETWEEN - Multiple in-betweening - create a series of points messages that changes in equal steps from firstmsg to lastmsg.
<DO> MBETWEEN, firstmsg, lastmsg, newmsg, steps <ENTER>

MCOPY - copy a series of messages from firstmsg through lastmsg (inclusive), from the fromdrive to thetodrive disk.
<DO>MCOPY, fromdrive, todrive [,firstmsg [,lastmsg [,newmsg]]] <ENTER>

SECTION - used to isolate the intersection of two or more overlapping objects.
<DO> SECTION <ENTER>

KEYSTROKE PROGRAMMING LANGUAGE (KPLs)

<COMMENT> This keys allows the user to notate the KPL program. The character(s) following <COMMENT> do not affect the program.

<COMPUTE> variable = value - where the value is computed and the result is placed into the variable.

<END> - stops the <LOOP>.

(INPUT-KEYB) - directs the KPL driver to let the operator type anything he/she wants to and will wait until the operator hits <END-OF-MSG> before resuming the program.

(LOOP) value - repeat the block of keystrokes and operations between <LOOP> and <END>, “value” times.

(LOAD-CNTR) variable/value - load the counter with the specified value.

(SAVE-CNTR) variable - save the number in the counter in the indicated variable.

(WAIT) value - Normally the KPL driver executes the program as fast as it can. When the driver encounters a <WAIT> operation it stops for the “value” fields before continuing. If the value is negative, the KPL driver will pause until an external trigger is received.
Appendix C
PRINTOUTS OF KPL PROGRAMS

The following pages are printouts of the KPL programs used to create this film.
COUNT MESSAGE TYPE SIZE NAME
--- ------ ----- ---- ----
   7 1.0  FPL PROGRM 149  SR/SA

[COMMENT] WEAVE SER OVER SER

[MESSAGE] FG SER? [INPUT KEY] [SAVE CNTR] F

[MESSAGE] BG SER? [INPUT KEY] [SAVE CNTR] B

[MESSAGE] HOW MANY [INPUT KEY] [SAVE CNTR] L

[MESSAGE] RECORD WHERE? [INPUT KEY] [SAVE CNTR] R


SEL BGD => WEAVE [LOAD CNTR] R [RECORD] [RECORD] [SAVE CNTR] R

[END]
<table>
<thead>
<tr>
<th>COUNT</th>
<th>MESSAGE</th>
<th>TYPE</th>
<th>SIZE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td>117</td>
<td>145</td>
<td>ST/SA</td>
</tr>
</tbody>
</table>

(COMMENT) WEAVE SER OVER STAT

(MESSAGE) STAT? (INPUT #B) [SAVE CNTR] F

(MESSAGE) FG SER? (INPUT #B) [SAVE CNTR] B

(MESSAGE) HOW MANY? (INPUT #B) [SAVE CNTR] L

(MESSAGE) RECORD WHERE? (INPUT #B) [SAVE CNTR] R


(END)
<table>
<thead>
<tr>
<th>COUNT</th>
<th>MESSAGE</th>
<th>TYPE</th>
<th>SIZE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>110</td>
<td>FPL PROGRAM</td>
<td>137</td>
<td>3R/SP ALT</td>
</tr>
</tbody>
</table>

[COMMENT] WEAVE SER OVER SER AND ALT RECORD

[MESSAGE] FG SER^ [INPUT KEYB] [SAVE CNTR] F

[MESSAGE] BG SER^ [INPUT KEYB] [SAVE CNTR] B

[MESSAGE] HOW MANY? [INPUT KEYB] [SAVE CNTR] L


SEL BGD < WEAVE > ALT < RECORD >
<table>
<thead>
<tr>
<th>COUNT</th>
<th>MESSAGE</th>
<th>TYPE</th>
<th>SIZE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>112</td>
<td>KFL PROGRAM</td>
<td>118</td>
<td>ST/SF ALT</td>
</tr>
</tbody>
</table>

(COMMENT) WEAVE STAT OVER SER

[MESSAGE] STAT\(^
\) [INPUT E\(\times\)B] (SAVE CNTR) F

[MESSAGE] EG SER\(^
\) [INPUT E\(\times\)EY] (SAVE CNTR) B

[MESSAGE] HOW MANY? [INPUT E\(\times\)B] (SAVE CNTR) L


<SEL BGD> WEAVE <ALT> RECORD

(END)
<table>
<thead>
<tr>
<th>COUNT</th>
<th>MESSAGE</th>
<th>TYPE</th>
<th>SIZE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>LG1</td>
<td>FPL PROGRM</td>
<td>171</td>
<td>WV SR/DR</td>
</tr>
<tr>
<td>COUNT</td>
<td>MESSAGE</td>
<td>TYPE</td>
<td>SIZE</td>
<td>NAME</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>-------------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>13</td>
<td>150</td>
<td>HFL FROGRM</td>
<td>98</td>
<td>DIGITIZE</td>
</tr>
</tbody>
</table>

MESSAGE: DIGITIZE FROM EDIT SUITE

COMPUTE: B=2500

LOOP: 999 (LOAD CNTR) 42 READ BGD

WAIT: -1 < DO > DIG.3 ENTER (LOAD CNTR) B = RECORD RECORD [SAVE CNTR] B

END}
COUNT  MESSAGE  TYPE  SIZE  NAME
--------  -------  -----  ----  ----
   6      114  PROGRM   148  SECSA/ST

[COMMENT] SECTION SER OVER STAT

[MESSAGE] STAT? [INPUT KEYB] [SAVE CNTR] S

[MESSAGE] SER? [INPUT KEYB] [SAVE CNTR] A

[MESSAGE] HOW MANY? [INPUT KEYB] [SAVE CNTR] L

[MESSAGE] RECORD WHERE? [INPUT KEYB] [SAVE CNTR] R

  <SEL-BGD> <DO> <SEC ENTER>

[LOAD CNTR] R <RECORD> <RECORD> [SAVE CNTR] R

[END]
COUNT MESSAGE TYPE SIZE NAME

0 11:1 FPL F9OGRM 99 MOVE SER

(COMMENT) MOVE SER

(MESSAGE) FIRST PIC7 [INPUT KEY8] [SAVE CNTR] A

(MESSAGE) HOW MANY? [INPUT KEY8] [SAVE CNTR] L 'SEL BGD'


<SHIFT DOWN

ALT: 'RECORD'

[END]
COUNT  MESSAGE  TYPE  SIZE  NAME
------  ------  ----  ----  ----
    1     100  FPL PROGRAM  101  REVERSE

[COMMENT] REVERSE A SER

[MESSAGE] FIRST PIC [INPUT KEYB] [SAVE CNTR] A

[MESSAGE] HOW MANY? [INPUT KEYB] [SAVE CNTR] L


[EOB] R

ALT - RECORD

[END]
COUNT MESSAGE TYPE SIZE NAME

5 113 KPL PROGRM 297 CHNGECOLRS

[COMMENT] CHANGE COLORS TO DIFF PALLETT POSITION <CEG>

[MESSAGE] BEGIN PIC? [INPUT KEYB] [SAVE CNTR] A

[MESSAGE] HOW MANY COLORS TO CHANGE? [INPUT KEYB] [SAVE CNTR] L

[MESSAGE] HOW MANY PICS? [INPUT KEYB] [SAVE CNTR] P [COMPUTE] *=:0


[COMPUTE] Z[X+1]=W [SAVE CNTR] Z[X+1] [MESSAGE] CHANGE TO


[COMPUTE] X=X+1 [END] ALT RECORD [END]
<table>
<thead>
<tr>
<th>COUNT</th>
<th>MESSAGE</th>
<th>TYPE</th>
<th>SIZE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1:1</td>
<td>KPL PROGRM</td>
<td>216</td>
<td>COLORING</td>
</tr>
</tbody>
</table>

[COMMENT] COLORING PICS IN FIXER

[MESSAGE] BEGIN PIC" [INPUT KEYB] [SAVE CNTR] A

[MESSAGE] HOW MANY PICS? [INPUT KEYB] [SAVE CNTR] P

[MESSAGE] HOW MANY COLORS? [INPUT KEYB] [SAVE CNTR] L [COMPUTE] X=0


[COMPUTE] Z(X+1)=W [SAVE CNTR] Z(X+1)

[COMPUTE] X=X+1 [END] [FONT FIXER]


[LODP] L [LOAD CNTR] Z(X+1) [FER] [INPUT KEYB]

[COMPUTE] X=X+1 [END] [INPUT KEYB] [ALT] [RECORD] [END]
**COUNT** | **MESSAGE** | **TYPE** | **SIZE** | **NAME**
--- | --- | --- | --- | ---
10 | 133 | FPL PROGRAM | 241 | CHANGE SET

**COMMENT** CHANGE SETS IN A POINTS MSG

[INPUT KEYB] [SAVE CNTR] L


[LOAD CNTR] 100 <ALT> <REC MARK> <TAB> <ALT> <REC MARK>

[LOAD CNTR] 75 <ALT> <TAB> [LOAD CNTR] 100 <ALT> <REC MARK> <TAB> <ALT> <REC MARK>


[LOAD CNTR] 28 <ALT> <TAB> [LOAD CNTR] 200 <REC MARK> <ALT> [RECORD] [END]
<table>
<thead>
<tr>
<th>COUNT</th>
<th>MESSAGE</th>
<th>TYPE</th>
<th>SIZE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>136</td>
<td>FPL PROGRAM</td>
<td>170</td>
<td>SEP SETS</td>
</tr>
</tbody>
</table>

[COMMENT] DELETE ALL BUT ONE SET AND MAKE PIC

[LOAD] L [LOAD CNTR] A • READ BGD • •• [SAVE CNTR] A
[LOAD CNTR] S • ALT • TAB • ALT • DELETE ROW
[LOAD CNTR] R • RECORD • RECORD • •• [SAVE CNTR] R
[END]
Appendix D
BIBLIOGRAPHY


Pritzker, Pamela. Magritte. New York: Leon Amiel Publisher, [no date].