PERSONALIZED MOVIES

by

Ralph J. Mayer

B. Sc. (Arch) McGill University 1974

B. Arch. McGill University 1975

Submitted in Partial Fulfillment
of the Requirements for the
Degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1979

Signature of the Author _____
Department of Architecture
June 19, 1979

Certified by _____
Nicholas Negroponte
Associate Professor of Computer Graphics
Thesis Supervisor

Accepted by _____
Professor Nicholas Negroponte
Departmental Committee for Graduate Students

Personalized Movies

Table of Contents

Personalized Movies

by

Ralph J. Mayer

submitted to the Department of Architecture on June 19, 1979 in partial fulfillment of the requirements for the degree of Master of Science.

Abstract

Personalized movies are used as an aid for instruction in maintenance and repair. Movies about bicycle overhaul are created by playing sequences from an optical videodisc which contains material filmed specifically for the project. Movies are personalized by means of an extensive profile describing each viewer's knowledge about the subject to be repaired and their preferences for presentation modes and teaching techniques. The computer model of the videodisc is constructed so that movies are interactive. The viewer can request that changes be made to the movie while it is being played, and those changes will appear without having to restart the movie.

Thesis Supervisor: Nicholas Negroponte

Title: Associate Professor of Computer Graphics

CHAPTER 1

Introduction

A personalized movie is a movie which has been prepared especially for you. It should contain only what you wish, or need, to see -- no more and no less. The storyline, the way in which the story is told, and the speed of the movie should all be attuned to your individual preferences, cognitive styles, and requirements. At least in theory, no two people should ever see the same movie. If another movie were prepared for you, it would be different from the first one, partially for variety, and partially because the viewing of the first movie changed you, and your needs in some way. A personalized movie would truly be one which is constructed uniquely for you.

One of the first questions which comes to mind about such a movie is: How does the movie maker know what kind of movie is best for me? The best way seems to be for you to tell the movie maker what you would like to see, and for the movie maker to prepare a movie based on that information. You could then watch the movie, and tell the movie maker whether this is satisfactory. The movie maker would make appropriate corrections. However, in order for you to be satisfied, the movie

maker must make some assumptions about what to show you; not only might you be incapable or unwilling to describe exactly what kind of movie you would like, but if you had to tell the movie maker exactly what you wanted to see, there would be no point in having the movie maker. If you tell the movie maker to change the type of movie which is playing, then the movie maker should not necessary start another movie. You should be able to continue watching the continuation of the current movie, but in a manner better suited to you. Thus a successful movie maker must be intelligent, interactive, and able to change movies as they are being made.

The concept of personalized movies is an answer to a very real problem currently being investigated by the Office of Naval Research. The armed forces in general, and the U.S. Navy in particular, have trouble properly maintaining their equipment due to an increasing amount of equipment and a decreasing pool of qualified repair personnel. The shortage of qualified people is due to many reasons, including the cessation of the draft and the decreasing numbers of people entering the armed forces to acquire professional training. One method of alleviating this problem is on-the-job training for these maintenance personnel, whose experience and knowledge differ widely. Personalized movies about maintenance and repair are a way of providing this training.

Personalized movies are a marriage of two concepts under study at the Architecture Machine Group: idiosyncratic sys-

tems, and the merger of computer graphics and picture processing. Idiosyncratic systems attempt to personalize the man-machine interface by having computers adjust themselves to each user. Computer graphics and picture processing use storage devices to save and retrieve pictures, rather than recreating each picture as it is required. The joining of these two concepts was made possible in part by the advent of the optical videodisc.

The optical videodisc plays televisison images stored on a disc approximately the size of a long-playing phonograph record Each disc currently contains 54,000 frames, or one half hour of video, with two accompanying sound tracks. Information is stored as FM encoded video in a series of pits which form a continuous "track". The disc is read using a laser beam and a light sensing diode; where there is a pit, no light is reflected back to the diode, and where there is a pit, light is sensed. These rapid fluctuations form the FM signal which is then decoded into standard televsison. One video frame is recorded on each revolution of the disc; therefore the laser beam may skip back to replay a frame. This permits slow motion, reverse playback, and stop frames. The videodisc is also randomly accessible. Any frame may be accessed by its frame number (1 to 54,000), with a maximum search time of approximately five seconds. Thus it is possible to play and fit together sequences from any place on the disc. In short, the videodisc is a perfect way for the movie

maker to play its movies.

The investigation of personalized movies is broken up into three areas: The preparation of the source material for the movies, the generation of a movie for a particular viewer, and the interaction between the viewer and the movie maker. Preparation of the source material involves creating a videodisc which contains all the sequences which are to be played by the movie maker. The movie maker itself is a computer program which shows a movie by selectively playing sequences from the videodisc. The movie maker is concerned both with the cinematic problems of how to concatenate sequences, and with the dynamic tailoring of movies to each viewer. The movie maker also interacts with and models the viewer in such a way that the viewer can easily tell the movie maker the types of movies desired.

CHAPTER 2

Concepts of Personalized Repair

A. Types of Personalized Repair

Maintenance and repair was chosen as the subject for initial investigation of personalized movies. The repair process is generally sequential and easily defined, but also has sufficient variability that instructions can be personalized for each repair person. For example, a typical sequence is to first loosen a bolt, then remove the bolt, and then remove the washer. A novice might be taken through each individual step and told what to do with each part, whereas an expert would just be told to remove the bolt. Overhaul operations were chosen specifically because they are more straight forward than diagnostic repair. The overhaul process always proceeds from step A to step B and so on, whereas diagnostic repair involves making many decisions to identify the part to be repaired. It was felt that such diagnosis is another topic altogether.

A bicycle was chosen as the object to be overhauled. A bicycle is small enough in scope that its overhaul can be adequately covered using one side of a videodisc, or in about

half an hour. This is in contrast to other choices such as an automobile or an air conditioner. Nonetheless, bicycle overhaul embodies many techniques of maintenance and repair. Bicycle repair was also particularly suitable for being played on the videodisc. Since the front and back of a bicycle are similar, some sequences can be used to illustrate more than one operation, thus effectively increasing the number of operations which could be covered. Furthermore, bicycles are easily obtainable and their repair is well documented.

B. Interaction With the Viewer

After choosing the subject, the outward characteristics of the personalized maintenance and repair system were specified. The first step was to generate a taxonomy of presentation modes (see Table 1). These included live action films of bicycle repair, animation of the repair process, spoken explanations, written explanations, diagrams, exploded views (both live and drawn), stills, and computer generated graphics to highlight all these modes. All modes of presentation should be available so that the movie maker can create movies using those modes preferred by the viewer. This is one way in which movies are personalized. The other ways are the teaching style employed by the movie maker, and the knowledge level for which the movies are made.

There are actually three ways in which a viewer may use a personalized repair system. The first way is for the viewer to ask the system for instructions on repairing a specific part such as the front hub. The second way is for the viewer to ask for specific instructions, such as the order of the parts on the axle. The third way is for the viewer to peruse all the information in the system by requesting information about a general topic such as bicycle wheels, or perhaps even bicycles in general. This project concentrates primarily on the first method, whereby the movie maker prepares a personalized movie about a chosen subject.

Table 1.

Presentation Modes

Live Film
Synchronous Sound
Narration
Animation
Cartoons
Exploded Views
Sketches
Diagrams
Graphic Overlays
Highlighting
Computer Graphics
Written Explanations
Computer Synthesized Sound
Slow Motion
Reverse Motion
Freeze frames

In order for the movie maker to personalize each movie, it must know who is watching the movie, how much that viewer already knows about the overhaul being done, and which modes of presentation the viewer prefers. The movie maker identifies each viewer simply by asking the viewer for his or her name. The movie maker keeps a list of all these names, as well as a description of how much they know and what they like. The more information remembered about each user, the more personalized the movies can be.

The movie maker maintains a viewer module which specifies whether each viewer is right or left-handed, whether they prefer movies or exploded diagrams, and how much they know about each type of bicycle overhaul. It is necessary to keep track of each viewer's knowledge about each overhaul operation, since somone who has repaired many flat tires may not have the slightest idea of how to take apart a hub. Each time a viewer watches a movie some knowledge should be imparted, and therefore the knowledge of a viewer is generally increasing. However, this is not always true. If a viewer has not repaired a bicycle for six months, some knowledge will be forgotten. Knowledge also becomes progressively harder to obtain. Much more will be learned the first time someone is shown how to repair a hub than the tenth time. Hopefully, by the one hundredth time the viewer will know everything about hubs, and will not need any further instruction. The movie maker uses this constantly changing viewer profile to create

movies which are personalized for a specific person at a specific point in time.

Another issue is how the movie maker interacts with the viewer. When overhauling bicycles, one of the most important pieces of information is identifying the part of the bicycle to be overhauled. Probably the easiest way for a viewer to do this is to point to a picture of a bicycle. This is done by displaying a frame from the videodisc on a monitor which is covered by a transparent touch sensitive screen. When the viewer touches the screen, the movie maker can determine the closest part of the bicycle, and using a picture of that particular part, query the viewer whether that is really the part to be overhauled. In this way, the choice of topic is chosen visually, and does not require the viewer to have any a priori knowledge of the names of parts on a bicycle. This information is provided by the repair system.

While a movie is playing, the viewer can interrupt it at any time to indicate that something should be changed. Possible changes are to expand or condense the explanation, to change the mode of presentation, or simply to play the last sequence over because the viewer wasn't paying attention. Such changes are used by the movie maker in future movie preparation. However, it is not neccessary to go back to the beginning of the movie; the next sequences shown will simply reflect the updated knowledge or preferences of the viewer.

The viewer indicates preferences to the movie maker by

special controls. The most basic of these is an expand/condense button, which the viewer uses to indicate whether the explanation provided is too brief and not comprehensible, or whether it is too long and is becoming tedious. Similar controls can be used for various types of audiovisual preferences. For example, the viewer can indicate a preference for or against movies, stills, drawings, diagrams, exploded views, audio explanations, animation, displayed text, and so on. The movie maker attempts to provide the viewer with sequences of the preferred types whenever possible. However, the preferences of the viewer must be balanced against the available material, and the fact that the viewer must be given suitable instruction for the object to be repaired.

The instructions for each overhaul operation must contain certain components. The foremost is a procedure for the overhaul. In addition, there should be an expanation of the function of each part in the assembly, how the parts fit together, and how the whole assembly functions. Explanations must be available for novices, experts, and people inbetween. For each operation it is also necessary to have an introduction to the operation, and a list of all the tools needed to perform the operation.

For those viewers who do not know how, there should be an explanation on how to use the tools. The movie maker would instruct novices in tool use the first time a tool is required. If the viewer requested more explanation, then the

movie maker would continue to instruct the viewer in the use of the tool until the use of the tools was percieved to be understood. Although all movies contain a variety of instructions, any particular movie will contain only a small portion of the information on the videodisk. Only that information applicable to the current viewer's knowledge, preferred learning methods, and presentation modes will be a part of the personalized movie.

C. Functions of the Movie Maker

Having discussed the outward characteristics of the movie maker, the next topic is the construction of the movie maker. Some information can be obtained from previous research in the area of Computer Assisted Instruction. Reseach in this area has been done at M.I.T by the Laboratory for Computer Science.

Out of this existing work came some ideas on computer instruction of students, or in our case, viewers. Computer aided instruction can apply a variety of teaching methods (see Table 2). These include telling the student what to do, what not to do, explaining why something should be done, providing the background and letting the student figure out what must be done, and using an analogy or refinement of a known procedure.

Another area of Computer Aided Instruction research is the modeling of the viewer. It is necessary to know how smart each viewer is, how fast they learn, what type of teaching methods they prefer, and if they are being shown movies, what kind of movie presentation they prefer. It is also useful to know what they dislike, as well as what they like. It is possible to model viewers using some predefined assumptions and a few basic questions, such as how well they feel they can repair a bicycle. Then the interaction between the movie maker and the viewer becomes very important in augmenting and changing the viewer model. If the viewer asks for expansion or condensation of a topic, then the computer model of the

Table 2.

Teaching Methods

Show Someone Else Doing It

Show the Assembly

Describe the Function of Each Part

Describe How it Works

Show Exploded Views of the Assembly

Procedural Rules of Thumb

Expand on Existing Knowledge

Analogy to Existing Knowledge

Show How Not to Do It

Library of Common Techniques

viewer's knowledge and learning rate should change. If the viewer asks for expansion or contraction again, then the computer model will change more drastically. Since it is not getting the same type of feedback as does a system which tests responses to specific questions, the movie maker's model of the viewer is based on very tenuous information. It is therefore very important for the movie maker to get specific requests to change the movie, and thus update the viewer model. Viewer models can also be examined for group behavior. If the previous ten viewers have asked for amplification on a particular subject, then it makes sense to give the next viewer that amplification straight off.

The movie maker must also have a model of all the information on the videodisc. The models of the viewer and the videodisc must be structured so that preparation of movies is done as they are being played and the movie maker can adjust immediately without having to create a new movie. The basic method for accomplishing this is to structure the model of the videodisc as a tree. Each node on the tree corresponds to a sequence or still frame on the videodisc. Selective expansion of the tree, based on each viewers knowledge and preferences, will generate personalized movies. This is explained in more detail in Chapter 4.

The movie maker plays movies by operating a videodisc system under computer control. It can instruct the videodisc player to move to a particular frame, to play forwards or

backwards, and to play the sequence at a specified speed. A videodisc system must contain at least two videodisc players with identical videodiscs. This is so that while one videodisc is in play mode, the other videodisc can already be searching for the next sequence. Another use of multiple videodiscs is that one videodisc can be playing sound while another videodisc is playing a visual sequence from a different spot on the videodisc. However, if both videodiscs are playing simultaneously, then there will be a pause while the next sequence is searched for. More than two videodiscs can therefore be used to provide additional flexibility and ensure continuity.

There are many different ways to combine the outputs from multiple videodiscs. As mentioned above, one can alternate between two videodiscs to provide a continuous picture made up of stills or film. Either sound track can be turned on and off as required since pauses in sound are not at all objectionable. Two videodiscs can also be played simultaneously to present material in different ways. Sound can be played from one part of the disc while another videodisc is displaying a picture from another place on the disc. Thus a viewer could get an explanation which was briefer, longer, or simply different. While one videodisc is playing a long sequence another videodisc can be used to temporarily replace the picture with a closeup of the part being handled. Significant operations can be replayed in slow motion, or perhaps backwards, to

illustrate a point. Computer generated graphics such as highlighting or arrows can be overlaid on top of the videodisc image to point out a tool or a part. Computer generated text can be displayed either to indicate a part or to amplify the spoken narration. The movie maker chooses between these display techniques to play a movie containing various audio and visual components from the videodisc.

In order to generate personalized movies of bicycle repair a videodisc with all the different sequences which might be used in a movie was manufactured. Material for the videodisc was created by filming the overhaul of a bicycle. Since the movie was to be prepared and edited by the movie maker program, the filming was somewhat different from traditional training films. In addition to live film footage there were still frames and exploded views taken from books on bicycle repair. A narration to describe both the bicycle and its overhaul was also included. Because of the difficulty in producing animation none was included. Written explanations and graphic overlays were also not put on the videodisc since they can best be stored digitally, and the videodisc for this project did not contain a section with digital encoding. Therefore, digital information was stored in the computer. However, in the future it should be possible to store such digital information on the videodisc itself. This videodisc is the source of material for the movie maker.

Computer programs were written to perform the functions

of the movie maker. The two major programs are the Editor, used to create the database model of the videodisc, and the Movie Player, which interacts with the viewer and creates personalized movies. The next two chapters describe the work done to create the personalized movie system. Chapter 3 describes the creation of the videodisc, and Chapter 4 describes the computer programs which make up the movie maker. The system is not yet complete, as some of the design criteria are only partially satisfied. However, the system was constructed with all these criteria in mind and its behavior is similar to that of the movie maker described here.

CHAPTER 3

Creating the videodisc Film

A. Filming

The filming was done in a private film studio. A set was constructed, consisting of a bicycle stand, a work bench, and backdrop. A bicycle expert was brought in to be the actor and overhaul the bicycle. The filming was done with 16mm film in order to obtain a high quality print. In addition, 35mm cameras were used to take still shots of the tools used for each operation, as well as each part after it was removed from the bicycle. These included closeups of individual nuts and ball bearings. A script was written which detailed all of the shots required to completely document bicycle overhaul. Included were shots from different angles and distances of all the overhaul operations. Most of the shots were designed to be from the point of view of the person actually doing the repair. This is a common training film technique, since there is a large difference between watching someone else take something apart and imagining yourself doing it.

Other techniques were devised to take advantage of the computer controlled videodisc. The selectivity of the movie

maker also permitted the inclusion of many different repair methods. Since a particular viewer would not be shown all of the possible repair techniques, it was possible to include information which might interest only a minority of potential viewers. This approach simply cannot be taken in conventional film making, where the material included should appeal to all viewers.

It was intended to film operations being done first by a right-handed person, and then by a left-handed person. Since right and left-handed people do operations differently, the duplication would ensure that everyone could see how they should place their hands to move the parts. However, due to time considerations and the difficulty of finding a left-handed bicycle expert, only one fairly short operation was filmed both from the left-handed and right-handed point of view.

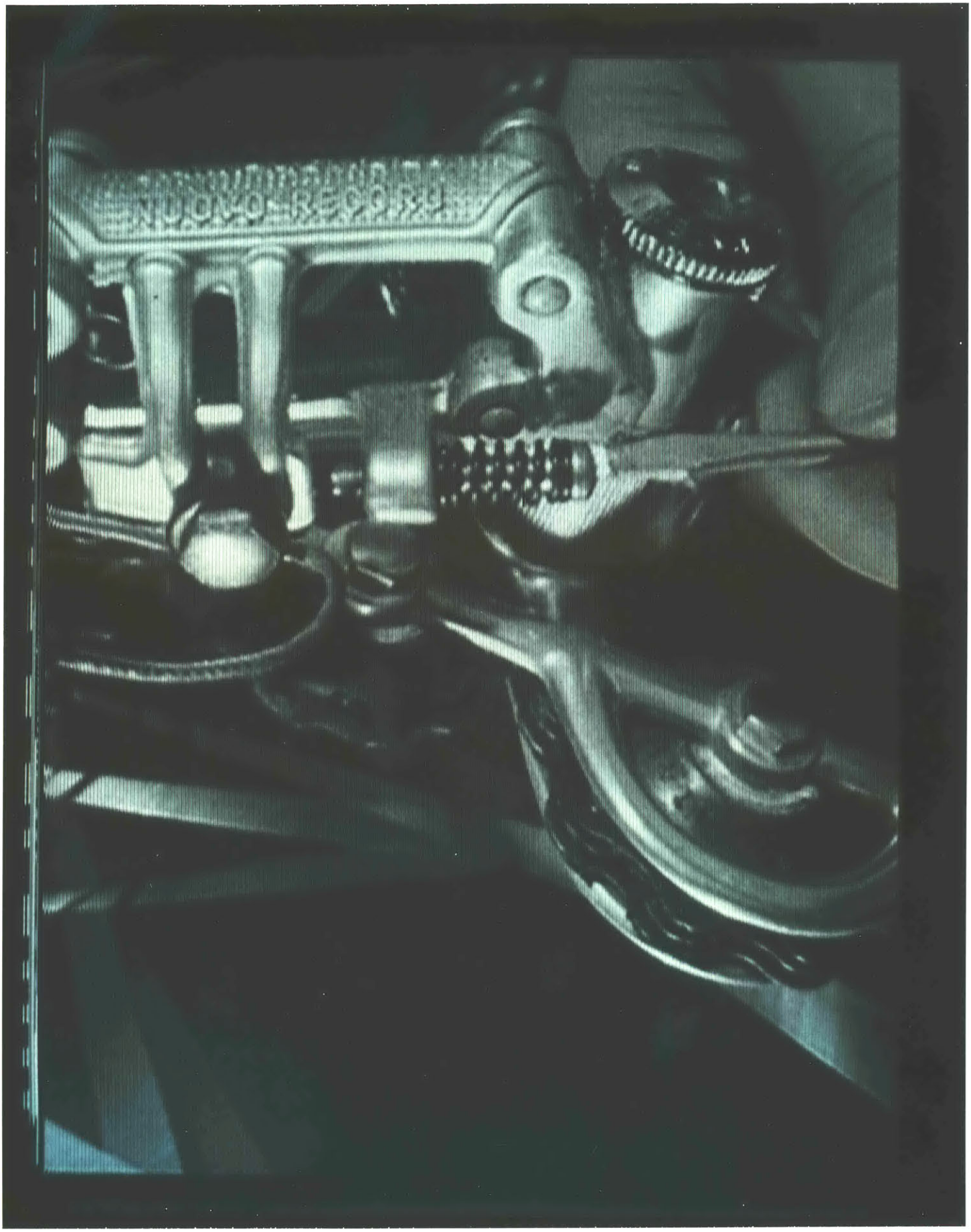
Most of the filming was done with synchronous sound, that is sound which is recorded and played back at the same time as the film. Such sound not only gives presence to the film, but is also an important part of the repair process. For example, it is very reassuring to hear the clang of a hammer, as a dustcap is tapped into place. Some procedures, such as the adjustment of derailleurs, can be done by sound alone. This is done by listening to how the chain sounds as the bicycle is shifted from one gear to another.

The script also included shots on how to use tools and

how to perform common procedures, such as cleaning bearings, which are a part of many bicycle overhaul operations. This set of shots was referred to as the "library". The library included shots of how to use each of the tools used in the filming. These descriptions show the correct way to use each tool and various techniques to take full advantage of the features of each tool. The cleaning procedures show how bicycle parts in general should be cleaned, using solvent to remove all grease and dirt. Throughout the film, care was taken to use the correct procedure.

While preparing the script, an attempt was also made to find alternate methods for performing various operations. It was felt that one would teach different techniques to an expert and a novice. However, it appears that although experts and novices probably do not repair bicycles in the same way, there is a "best" technique which should be taught to everyone. The differences are mostly in the way in which techniques are explained. It is reasonable to tell an expert to tighten a nut "until it is hand tight". A novice needs a more quantitative description, such as, "until the wheel just stops spinning". A novice also needs information on how to determine whether a bolt is too tight or too loose. Illustrations of adjustment for both the expert and the novice are included for most of the overhaul operation in the film.

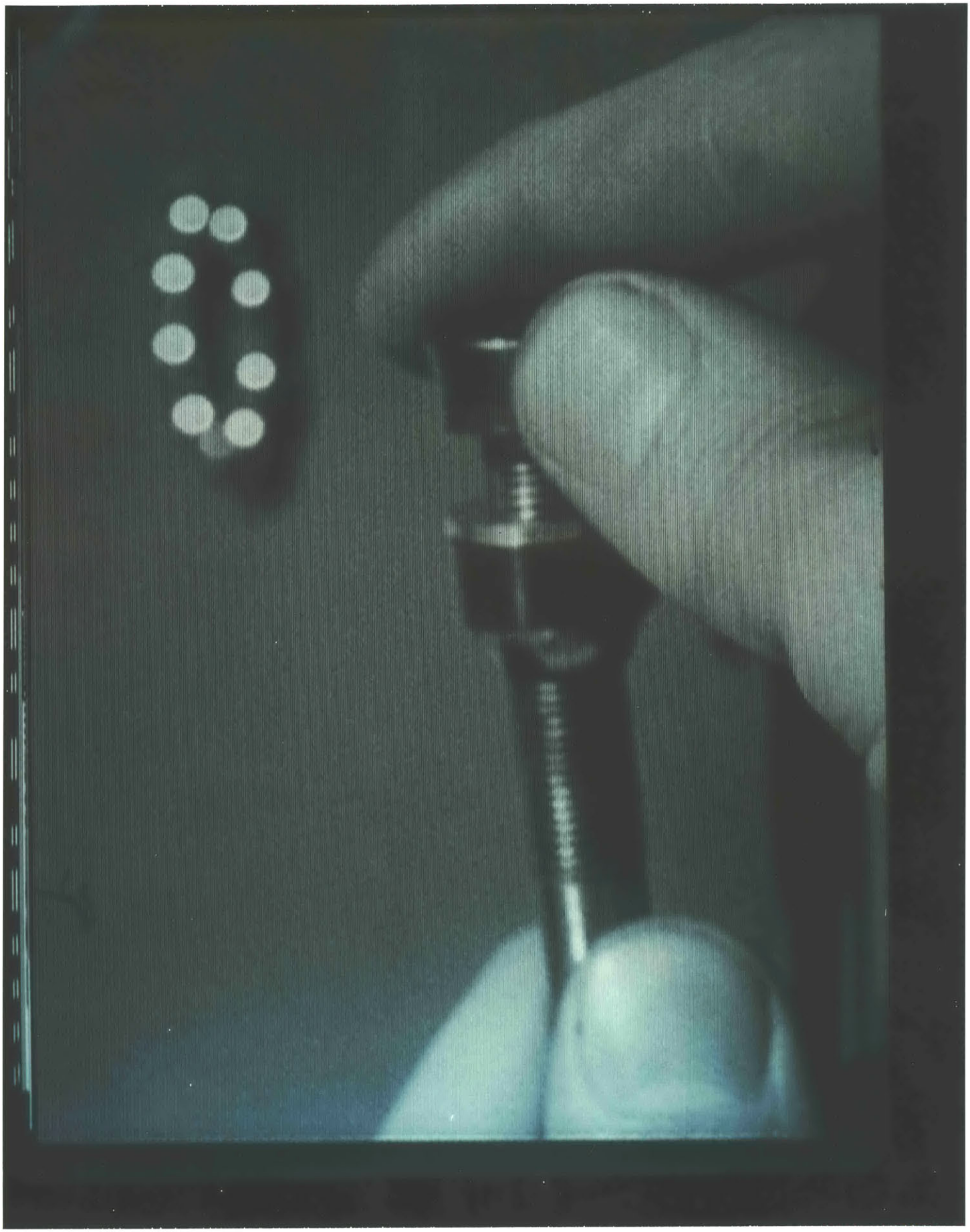
Another method of instruction which was developed during the filming was to illustrate how the parts fit together.



Good mechanics may be able to take any object apart once they understand how the assembly fits together. To instruct this type of repair person, shots were filmed of how the internal pieces on a bicycle are put together. For example, an axle and its locking devices are filmed being put together outside instead of inside the hub, thus illustrating how they must be taken apart.

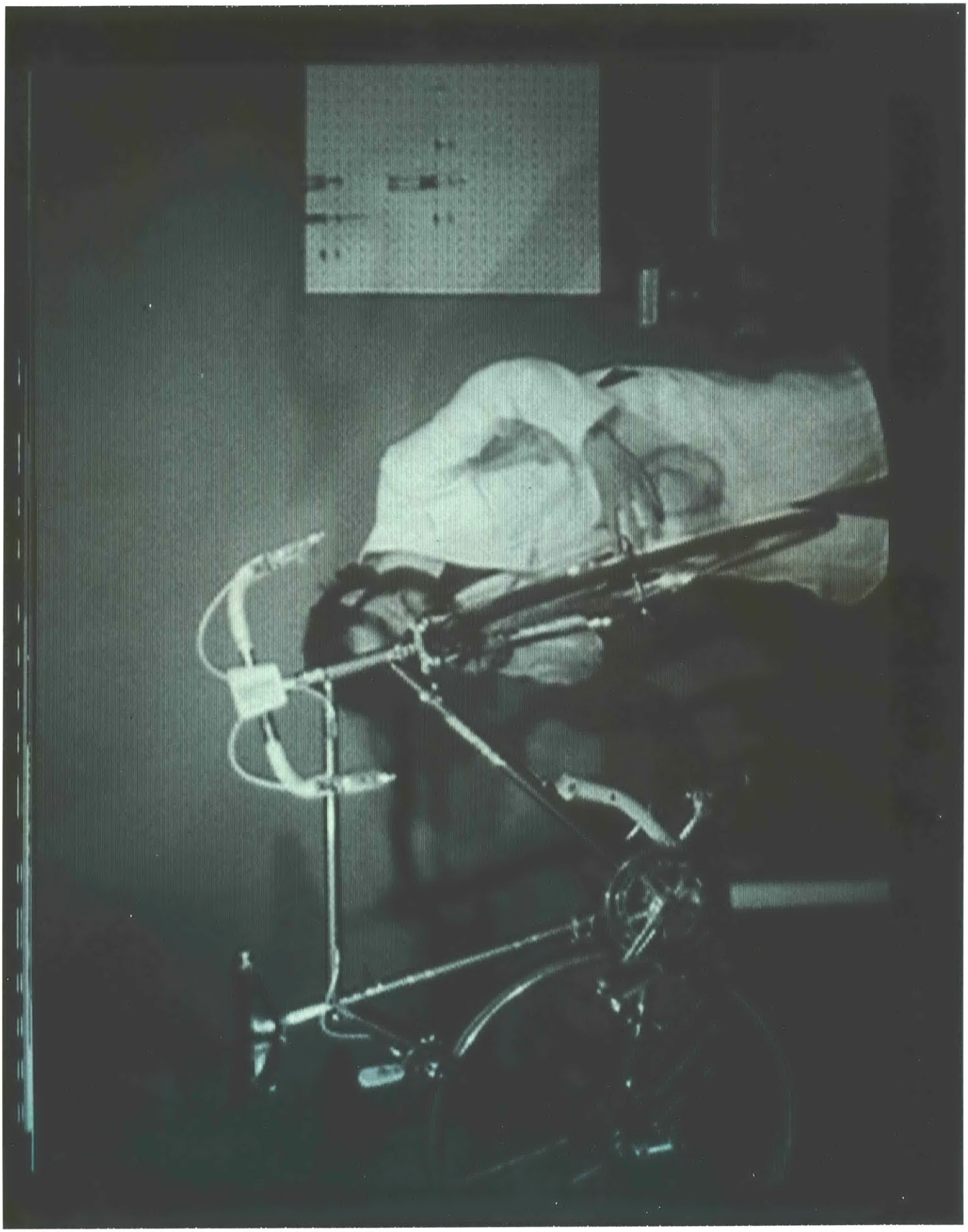
Two other similar types of instruction were also filmed at the same time. The first such method is to explain not how an assembly fits together, but how it works. To illustrate this mode of teaching, the function of important parts, such as locking devices, was demonstrated. Another teaching variation is to simply show all the parts and their relationship to one another. This is very similar to an exploded diagram, but uses an actual picture instead of a diagram. An experienced repair person can quickly understand how to disassemble an object once the individual parts are known.

Despite all these variations, the bulk of the filming was devoted to traditional methods of instruction. That is, showing the pupil a step-by-step procedure to follow. This was due in large part to the fact that performing each operation in its entirety is much more time consuming than simply describing the object to be overhauled. Because of the time required to film each operation, only a subset of the overhaul operations on a bicycle were filmed. These operations were filmed from various angles and distances. There were a number



of differences between this film and a traditional instructional film. Contrary to normal practice, closeups of operations were filmed in their entirety. This is so the movie maker may cut to a closeup at any time, and not be restricted to only those closeups which seemed appropriate while filming. During a personalized movie only a portion of the operation may be shown, and the type of closeups required may be different depending on exactly which portions of the operations are being shown to a particular viewer.

Similarly, the amount of detail required to create a movie may differ, since each type of shot shows different kinds of information: A long shot may show how the person should stand next to the bicycle, a medium shot would show how ones hand is placed on the part, a closeup would show how the part is being handled, and an even tighter shot is needed to show how the shape of the part determines how it must be handled. Since different viewers need different types of information, all these various levels of detail should be included. However, the number of shots required to satisfy any request is enormous. Therefore, only those shots which were felt to be fairly useful were filmed. Even so, there was a tremendous amount of information.



B. Cutting and Editing

After getting back the developed film and removing all the outakes, there were about one and a half hours of film describing somewhat over half of the possible overhaul operations on a bicycle. In order to cut the film down to half an hour, a lot of material had to be removed. First of all, much duplication was removed. For example, two views of the same subject which did not really have any content difference were each cut in half, so that half of the operation was seen from one angle and the other half from another angle. Although this is not ideal, the viewer does get to see both views and thus gets a sense of space, while freeing up valuable room on the videodisc. In addition, shots which were repetitive, such as the derailleur adjustment, were cut drastically. These shots can be intercut and replayed as necessary to recreate an illustration of a derailleur adjustment.

The closeups were carefully scanned, and cuts were made to remove those frames where no action was going on in the frame. For example, when a nut was being put down on the bench, there is no action in the frame. The movie maker can either still frame to simulate the time needed to put down the bolt, or can cut to a shot of the nut being placed on the workbench. Actually, in many instances watching the entire operation is rather tedious, and it may be desireable to have the movie maker occasionally skip over pieces of film such as

the setting down of nuts.

Some operations such as the overhaul of the front and rear hubs were very similar. Therefore, many of the duplicate shots were removed. The movie maker can use the same sequences to illustrate those parts of the operation which are identical. After removing all this film, slightly more than half an hour of film still remained. Some of the simple operations such as brake adjustments, which had been included more for completeness than for uniqueness, were discarded. This left a series of bicycle shots of just under half an hour.

In addition to the film itself, there were 35mm still shots taken of all the bicycle parts and all the tools used for each operation. This was augmented by pictures, diagrams, and exploded views from a number of highly regarded books on bicycle repair. In total there were about 500 still frames added to the film. This is enough to make a formidable slide show, but seems rather insignifigant compared to the forty thousand plus frames of film. This was the visual information for the videodisc.

The visual information had then to be laid out in some sensible fashion on the videodisc. Since search time varies proportionately with how far the videodisc must move, it is best to have all the shots which are likely to be seen in sequence very close together. Thus the videodisc layout was done on the basis of subject. The relationship of each subject is also important, since similar operations such as the

overhaul of the front and rear hubs have shots in common, and hence should be adjacent. Some operations must also be performed before other operations. For example, the front wheel must be removed before the headset can be disassembled, and therefore these subjects must also be adjacent. Finally, the library sequences and still shots may be shown as part of any subject, and therefore they are grouped in the middle of the videodisc. The final videodisc layout consists of a number of subjects grouped around the central resources. Within each subject, there are long shots, cutaways, and closeups. Watching the videodisc from start to finish is rather confusing; it is designed to be played by the movie maker, not to be seen in its entirety.

C. Narrating

A narration was then prepared to accompany the film. Although the narration had been worked on throughout, it was impossible to fix the narration until the cutting of the film had been finished. Since there are two sound tracks on the videodisc, there was a full hour available for narration. The two sound tracks were utilized as follows: One sound track was designated to follow the action on the screen, and the other sound track would contain additional information. The first sound track also included the synchronous sound recorded during filming. Sound had been recorded during all the long and medium shots, and a few of the closeups. Since the close-ups are generally overlapped with another shot, sound is not usually required. The layout of the sound tracks permits most shots to be played with synchronous sound and a voice over describing the operation being performed. The second sound track is generally played with either a still frame, or with a picture from another part of the disc. This requires two videodiscs, but even then there may be a pause while searching for the next sequence to play. Of course, with three or more videodiscs, this is not a problem.

As mentioned above, the narration for the first sound track was written to follow the action in the film. This meant that in some cases when a lengthy explanation was required it was necessary to move a part of the description to

the second sound track. In other cases, such as cutaways where no description is required, additional narration which is not played with the picture was included on the first track. A script was written to describe each operation as it was performed in the film. Trial run-throughs were made by reading the script while watching a videotape of the final film. The script was typed up using a computerized text editor which made it easy to modify and move information from one sound track to the other. It is also possible for the movie maker to generate text on the screen using the computer file of the narration. Run-throughs were made until the pace of this "play be play" narration exactly matched that of the film.

The one section of the film which did not lend itself to this type of description was the adjustment of the derailleurs. Unlike the other bicycle operations, the viewer could not comprehend what was being done simply by watching the silent film. For this section of the film the narration was written without regard to the picture, with the idea that the movie maker will fit the picture to the narration instead of the other way around.

In addition to procedural descriptions, there were many other things to say about bicycle overhaul (see Table 3). First of all, alternate procedures, and summaries had to be prepared for all operations. These narrations were at a different pace than the procedural descriptions, and were gener-

Table 3.

Types of Narration

Procedure to be Followed

Abreviated Procedure

Introduction to Job

Summary of Job

Overview

List of Tools and Materials Required

Functional Description of Part

Functional Description of Assembly

Summary

How the Parts Fit Together

How to Positon Yourself

Where the Work is Done (bench, stand)

Why Things Are Done in a Particular Way

Things to Check For

Hints

How Much is Left to be Done

Possible Errors

Results of Possible Errors

Glossary of Terms

Listing of Parts and Tools

ally placed with the cutaways and closeups for a particular operation. For each operation there was also an introduction and a list of all the tools needed to perform the overhaul. There was a description of each part and how it fit into the assembly, as well as an overall description of the assembly and how it functioned. There were additional hints and explanations, such as suggesting how to test whether a bolt was tight enough, and what might happen if it wasn't. There were comments on why certain proceudures should be done, the sequence of operations to be followed, which operations were upcoming, and how many operations were left to be done. There was also a glossary containing the names of all the parts and tools shown on the videodisc. The movie maker can use the glossary to say the name of a part while showing a picture of it to the viewer.

All of these alternate and additional explanations filled up most of the second track as well as the unused space on the first track, thus utilizing all of the audio portion of the videodisc. The actual recording was done while viewing the videotape of the film, ensuring that the timing of the narration matched that of the film. The two sound tracks were recorded scene by scene, and then cut to exactly match the film. As in the film, all remarks about a particular topic were grouped together. The first narration which contained the play-by-play description was mixed with the synchronous sound. Then the two sound tracks, and the picture, were sent to California to be manufactured into a videodisc.

CHAPTER 4

Components of the Movie Maker

A. The Database

Once the film was on the videodisc, a database had to be designed to describe the film to the movie player. The database describes the contents of the videodisc, both in terms of cinematic considerations and bicycle repair. The cinematic portion is general enough to be used for any type of film; the bicycle repair portion is particular to this project. Information on the videodisc is broken up into segments. A segment corresponds to one shot. Thus a segment for a still frame is only one frame long, whereas a film segment for a long overhaul operation may be several thousand frames long.

Each segment is described by a descriptor record (see figure 1). A segment descriptor record contains not only the physical disc position of the segment (start and stop frames), but also cinematic data. This includes the cinematic type (audiovisual, audio only, visuals only, or still), the angle of view, and the degree of zoom (long shot, medium, closeup). There is also data on whether the shot is for right-handed people, left-handed people, or both, how much detail is shown by

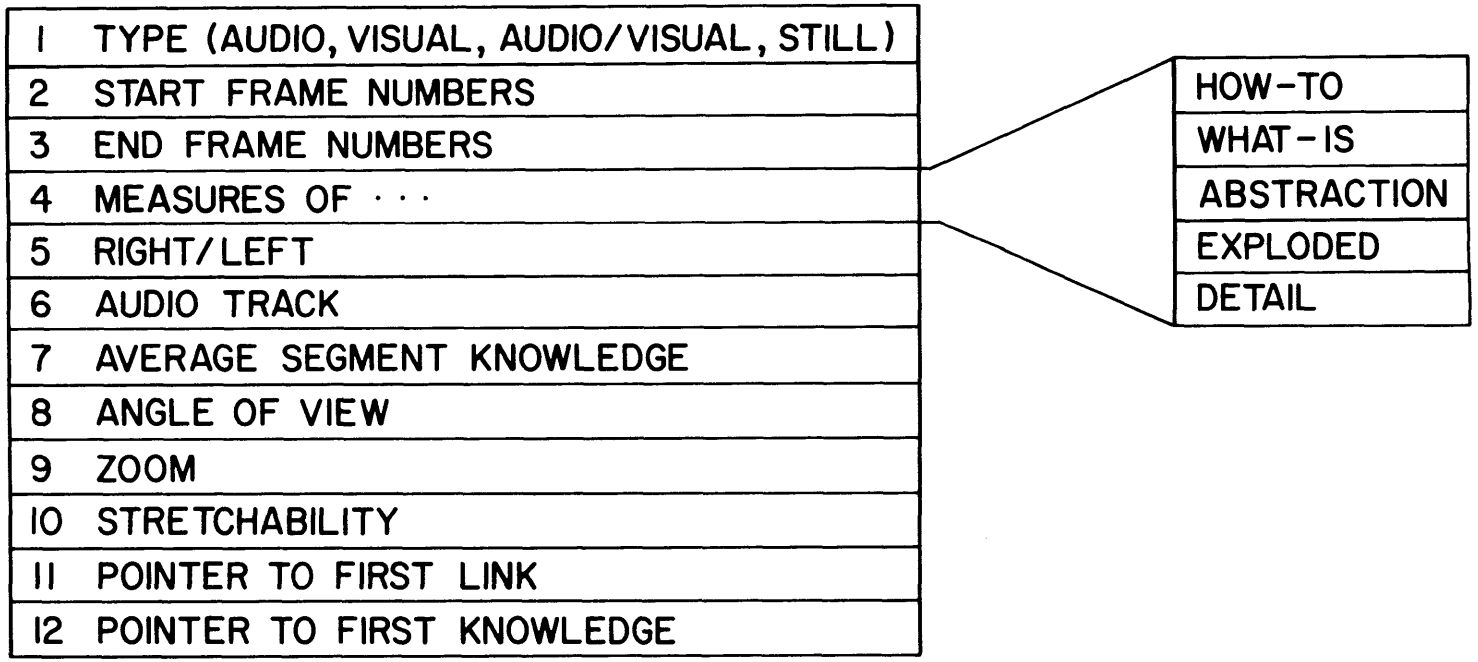


FIGURE 1 SEGMENT DESCRIPTION RECORD

the shot, whether it is an exploded view, and how much of the content is procedural or descriptive. The distinction of whether the mode of teaching tells the viewer how to do the operation, as opposed to describing the function and operation of the the parts and the assembly, is referred to as"how-to" and "what-is" respecitvely. This distinction is the way in which different teaching modes are recognized by the movie player.

Associated with each segment descriptor record are a number of other descriptor records. Their relation to the segment are shown in Figure 2. The first of these marks a subject descriptor record (see Figure 3). The subject descriptor record describes the function of the segment, the name of the operation being illustrated, and how much of the operation is shown. The function of segments include, for example, an introduction to a topic, a description of how to use a tool, a long shot of the entire operation, or a closeup. The name of the operation, such as the hub, is used to associate all the segments which describe the overhaul of hubs. The amount of the operation described by the segment is specified by an estimated percentage of completion at the start and end of the segment. This is used to match up major sections of the operation, so that the appropriate closeup can be overlapped onto a long shot of a bolt being removed.

In addition to the subject descriptor record there are a number of sub-subject descriptor records (see Figure 4). Each

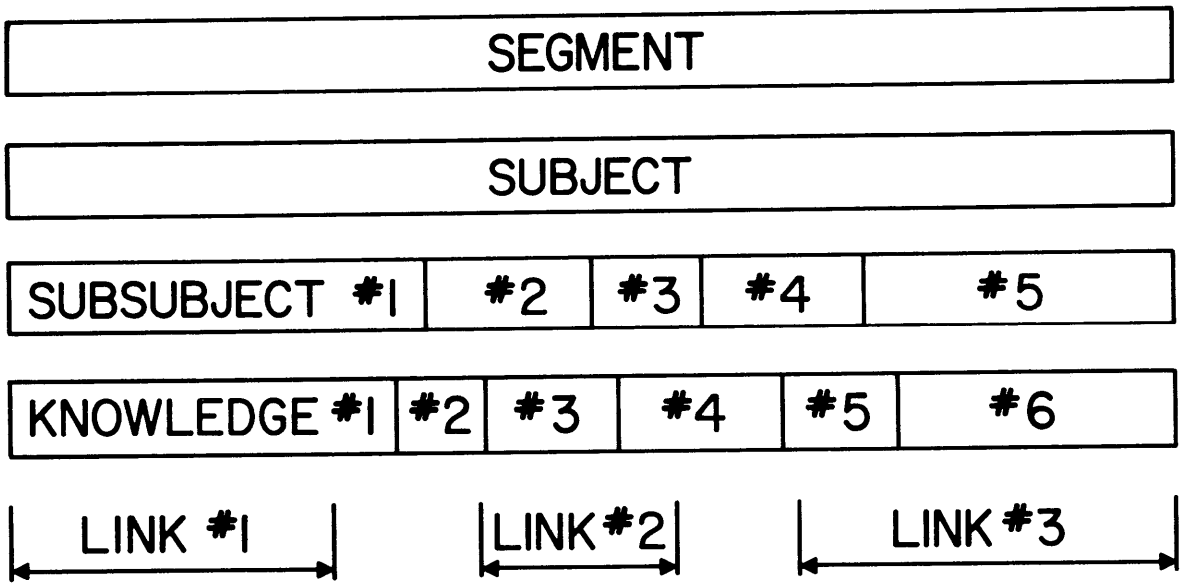


FIGURE 2 RELATIONSHIP OF RECORD TYPES FOR A SEGMENT

1 TYPE (MASTER, DETAIL, CLOSEUP, ETC.)
2 SUBJECT NAME
3 START % COMPLETION
4 END % COMPLETION
5 POINTER TO FIRST SUBSUBJECT

FIGURE 3 SUBJECT DESCRIPTION RECORD

1 TYPE (ALWAYS 1)
2 START FRAME NUMBERS
3 END FRAME NUMBERS
4 LOCATION (FRONT, BACK)
5 PART NAME
6 TOOL NAMES (2)
7 OPERATION (APART, TOGETHER)
8 SUBASSEMBLY USED
9 POINTER TO NEXT

FIGURE 4 SUBSUBJECT DESCRIPTION RECORD

of these describes one step within the segment, such as the removal of a single part such as a bolt. There must be at least one sub-subject descriptor per segment, and there may be as many as required. A sub-subject descriptor record contains information on which part is being removed or installed, whether the segment shows it being removed or installed or both, whether this sub-subject applies to the front or the back of the bicycle or both, which tools are required, and what other operations or subassemblies must be included as part of this operation. An example of an included operation is that the front wheel must be removed before overhauling the headset or the front hub, or repairing the front tire. The front wheel removal is therefore considered a subassembly which must be done as part of these other operations.

There are assumed to only be two tools per sub-subject since people only have two hands, and can only handle two tools at a time. If a third tool is used, then the subject can be subdivided further. Like the subject description record, the sub-subject description record also contains the percentage of completion at the begining and end of the sub-subject. Once again, these percentages are used to control the overlapping of segments which describe the same operation. They are used by the automatic linker which is part of the editor and is described below. It is also necessary to specify exactly where the sub-subject appears within the segment. In order that this information can easily be verified

on the videodisc, the sub-subject position is stored using absolute videodisc frame numbers, just as in the segment descriptor.

In the same way as the subject is subdivided into sub-subjects, each segment is subdivided into cutable scenes. To accomplish this, each segment is associated with one or more knowledge descriptor records (see Figure 5). Knowledge descriptor records serve two purposes. Firstly they divide a segment which has narration into short pieces, each of which has one or two sentences. This enables the movie player to cut in and out of a sequence without chopping it off in the middle of a sentence. Secondly each knowledge descriptor has, as its name indicates, the amount of knowledge associated with each portion of the segment. This information is used by the movie player to decide whether a particular portion of the segment should be played. There are two measures of knowledge for each segment portion. The first is how knowledgeable a viewer should be before this segment portion is shown. For example, a brief reminder would only be given to a viewer who had done the operation many times before; it would not be shown to a novice to whom it would just be confusing. The other measure of knowledge is how useful the sequence is likely to be to the viewer. The movie player would show only the highlights of an operation to a fast learner, whereas a slow learner would be taken through all possible elaborations. Like the sub-subject descriptor records, the knowledge des-

1 TYPE (SAME AS SEGMENT TYPE)
2 START FRAME NUMBERS
3 END FRAME NUMBERS
4 STOP FRAME NUMBER
5 KNOWLEDGE REQUIRED
6 KNOWLEDGE GIVEN
7 POINTER TO NEXT

FIGURE 5 KNOWLEDGE DESCRIPTION RECORD

criptor records contain absolute videodisc frame numbers to specifiy their position within the segment. In addition, a representative stop frame for the segment portion is specified. This stop frame is used by the movie player to illustrate the segment portion when continuous motion is not desired.

Lastly, each segment may also have link descriptor records (see Figure 6). Link descriptor records specify that another segment may be spliced onto the current one. A link descriptor record specifies the segment which is to be linked, and an exit and reentry position for the linked segment. The segment with the link descriptor record is referred to as the segment linked from, and the segment specified in the link descriptor record is refered to the linked segment. The exit and reentry positions may be anywhere within the segment containing the link (see Figure 7). They may be the same, in which case the linked segment is actually a diversion within the current segment. For example, to illustrate the removal of the front wheel before the disassembly of the headset, the segment for the headset disassembly has a link to the segment describing the removal of the front wheel. The link exit and reentry position are both at the start of the headset segment; the movie player will therefore shown the removal of the front wheel before the overhaul of the headset. In the case of a closeup, the exit and reentry positions would be before and after the shot being replaced by the closeup. For an abrevi-

1 TYPE (SAME AS SUBJECT TYPE)
2 EXIT FRAME NUMBERS
3 REENTRY FRAME NUMBERS
4 SEGMENT LINKING TO
5 POINTER TO NEXT

FIGURE 6 LINK DESCRIPTION RECORD

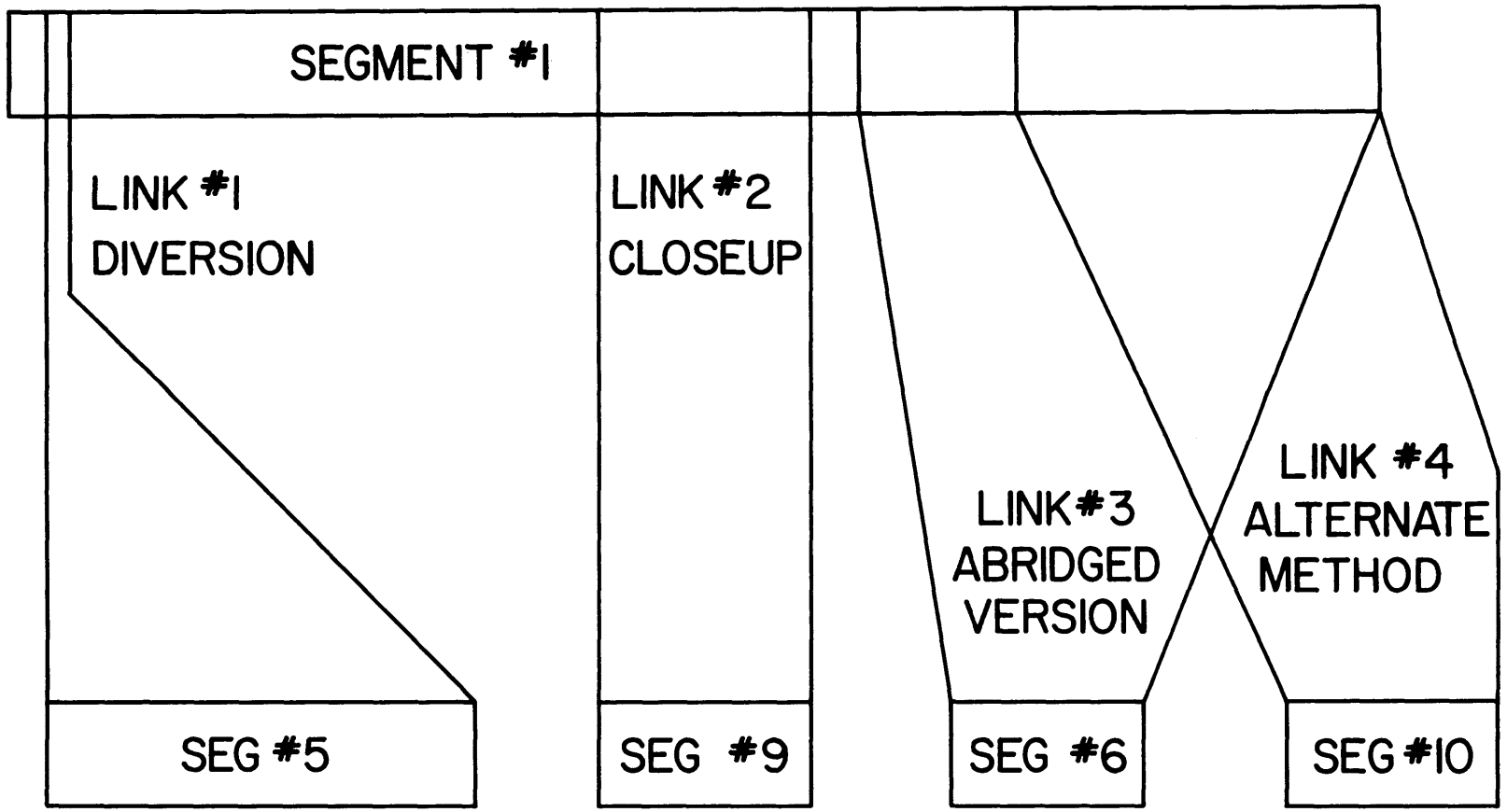


FIGURE 7 LINK USAGE

ated sequence, the link exit and reentry would be further apart, thus creating a shorter route between two parts of the original segment. As in all the other descriptor records, the exit and reentry positions are absolute videodisc frame numbers.

In order to create a database describing an entire movie, segments are linked together in a sort of hierarchy. One segment is linked to a second, which in turn is linked to a third, and so on. Thus the links form a tree, with each segment being a node. However, since a segment may be linked by more than one other segment, the structure is actually a graph (see Figure 8). Still, it is often usefull to think of it as a tree with the common nodes just being duplicated. Within this structure there is an implied hierarchy. The top level segments, called "master" segments are medium or long shots which describe the entire operation being performed. These segments link to those which show alternative ways of doing things, which in turn link to closeups, and so on. There are actually more intermediary levels, thus accomodating all the variations which exist on the videodisc (see Table 4). In addition, any level segment may link to a generally usefull segment such a closeup, or a description of how to use a tool. Master segments may also link to other master segments which are to be used as subassemblies. However, in general, segments only link to segments which are below them in the hierarchy.

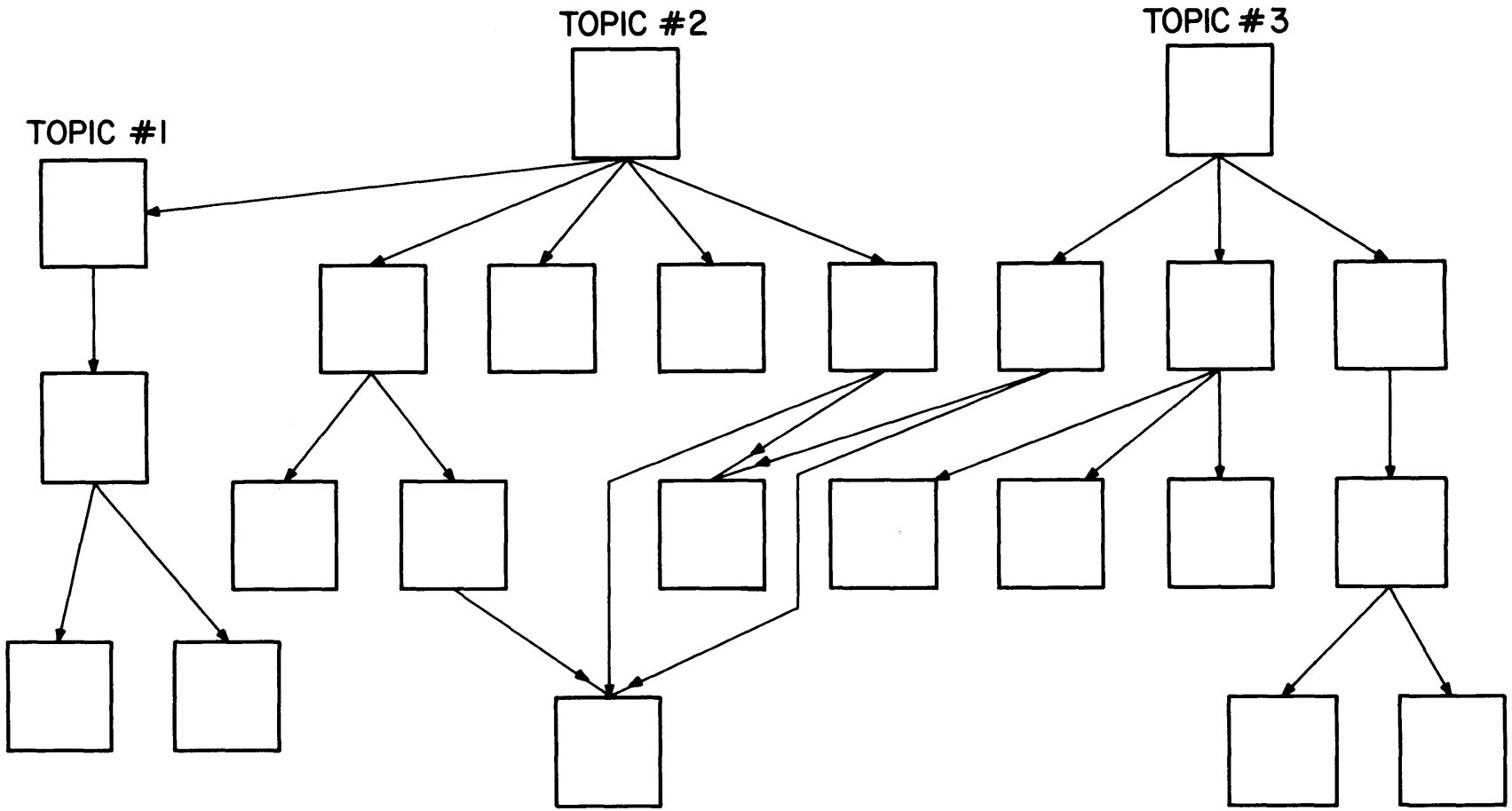


FIGURE 8 THE DATABASE GRAPH

Table 4.

Heirarchy of Segments

Master Segments

Sub-master Segments

Different Ways of Doing the Operation

Detail Shots

Cutaways

Closeups

Macro Closeups

Subassemblies

Tool Usage Shots

Introductions

Overviews

The physical layout of the database is as follows: Each type of descriptor record is kept in a separate file, or to use the MagicSix terminology, segment. Within each file there are multiple fixed length records. There is a count of the allocated records at the start of each file. This count is used when allocating new records and when setting the length of the file. The first field of each record contains the record type. This type, whose meaning varies from record to record, is set to zero when a record is deleted thus enabling the space to be reclaimed. The rest of the fields vary from record to record. Segment records are allocated as segments are created. Since there is exactly one subject descriptor record per segment, a subject record is created whenever a segment record is created. For each segment there may be a variable number of sub-subject, knowledge, and link descriptor records. The segment contains the record number of the first sub-subject, knowledge, and link descriptor records. Each such record then contains the number of the next record of that type, with a zero signifying the end of the chain for the segment (see Figure 9). Links also contain the record number of the segment which is being linked. Each file is defined as a very large array, so that no explicit calculations have to be done to find a particular record within a file.

It was mentioned above that the descriptor records contain information on which operation is being performed and which part is being handled. Obviously it is desireable to

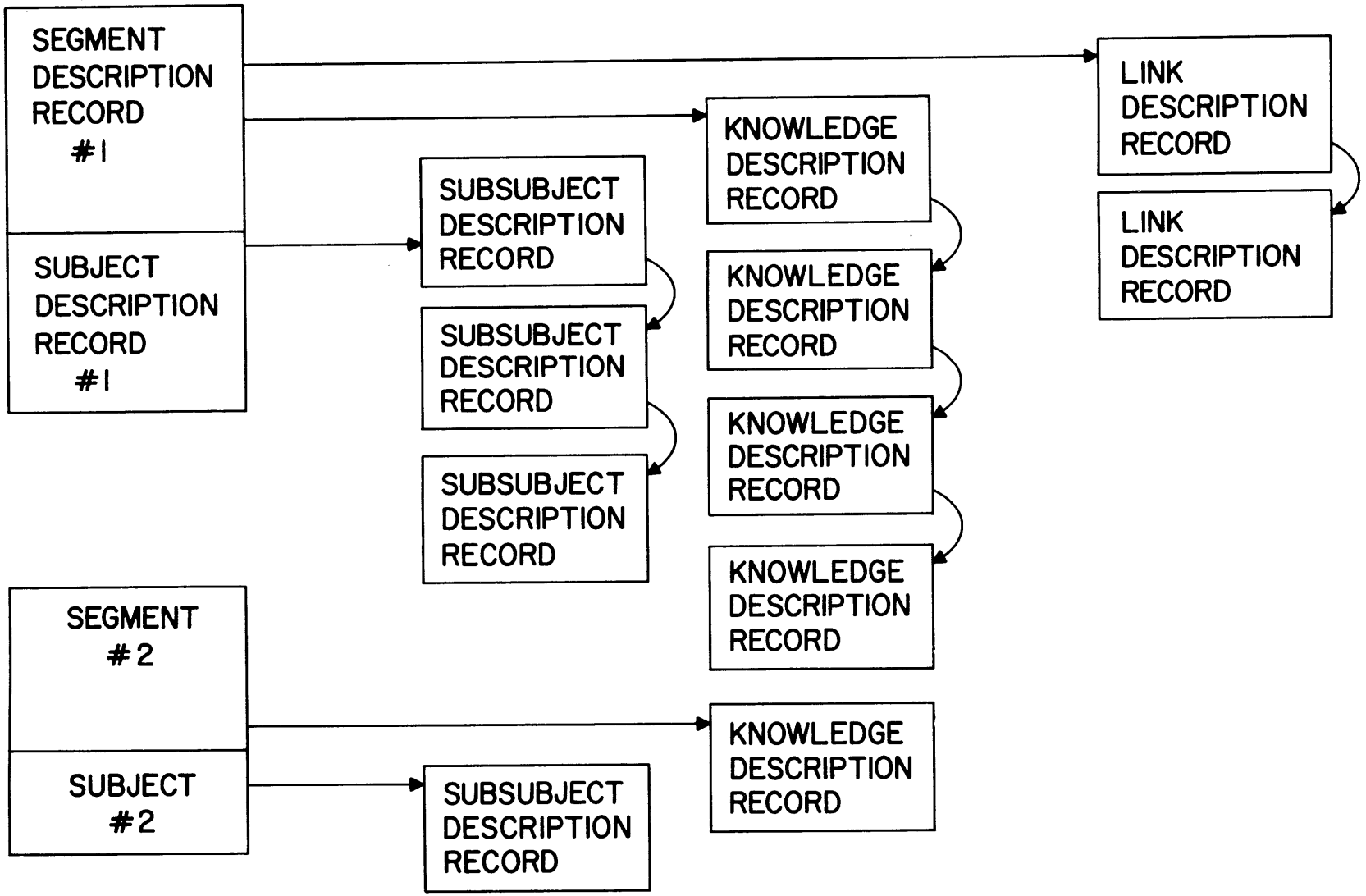


FIGURE 9 RELATIONSHIP OF DESCRIPTION RECORDS

have this information displayed in English, rather than having to remember that operation 1 is the headset, 2 is the hub and so on. On the other hand, storing all these characters many times in each file would cause the files to get unmanageably big. Therefore, the last part of the database is a dictionary which keeps a record of all the numberical types in the descriptor records and what they mean. This dictionary is updated automatically as new types are created. This is done by the database editor, the next topic.

B. The Editor

The database editor facilitates the entry and updating of the various descriptor records in the database. This editor is only suitable for editing the videodisc database, as it contains a great deal of knowledge about the type of information described by the database. The general mode of operation of the editor is as follows: it opens one field at a time within a given record, displays the old contents of the field, and prompts for the new contents. At this time one can enter either a new value, or any editing command. See Table 5 for a list of Editor Commands.

Editing commands permit one to move to any segment within the database. Once within a segment, one may go to any sub-subject, knowledge, or link descriptor record. The segment and subject descriptor records are viewed as a single record since there is a one-to-one correspondence between them. Whenever the editor moves to a new record, it first of all displays the entire record, and then opens the first field in the record. If a new value is entered, the editor moves to the next field in the record. It is also possible to move to the next or previous field or record. Records may be copied or deleted, and new ones may be added.

The editor keeps track of all the pointers from a segment to its sub-subject, knowledge, and link descriptor records, as well as the linked list of descriptor records of each type.

Table 5.

Editor Commands

a Add a new record

b go to Begining of record

c Copy a record

d Delete a record

e Error-go back to previous field

f Field-go to next field *

g Generate links automatically

j go to sub-subJect record *

k go to Knowledge record *

l go to Link record *

m Move to segment pointed to by link *

n Next record *

o Order the records for a segment

p Previous record *

q Quit

s go to Segment record *

t Type the entire database

u Up in tree (opposite of Move)

v View on videodisc

w Watch current frame

? "help" command

* means the command uses a positive integer argument.

The editor attempts to be helpfull by initializing values whenever a new descriptor record is created; for example, when a new knowledge record is created the start position of the record (its videodisc frame number) is calculated to be the end frame plus one of the previous knowledge record for the current segment. The end position is initialized to be the same as the end of the entire segment. Similar initializations are done for other record types. Assembled into the editor is information about which type of information goes into each field of each record type. There are five types of fields: videodisc frame numbers, numberical types which use the dictionary, numerical values which have a minimum and maximum, pointer fields which are created and updated by the editor, and unused or spare fields. The editor only prompts the user for fields of the first three types, and checks the legality of information put into each field.

As mentioned above, there is a dictionary containing the meanings of all the numerical types within the data base. Whenever the editor opens a field containing a numerical type, it prompts for either a character string or a number. If a character string is typed, the editor compares it with all the existing types for that field. If the entry already exists, the associated value is used. Otherwise the editor prints out all the existing dictionary definitions for that field and asks if a new entry should be made. If so, the dictionary is updated appropriately, otherwise an existing defintion can be

used. Whenever the contents of a field are displayed, both the numerical value, and its definition are shown. There are separate dictionaries for most fields in the database, although some fields share the same dictionary entries, since the fields have the same numerical types.

The editor also allows one to move from a link to the segment being linked, and so on down the tree. It remembers the path taken so that one can also go back up through the tree. In order to verify the database on the videodisc the editor will play any segment, sub-subject portion, knowledge portion, or link portion on the videodisc. If the start or end of the portion is incorrect, the appropriate frame number can be adjusted, and then the portion previewed again. When playing a link, the editor will play part of the current segment, followed by the segment being linked, and then by the current segment again. This allows one to preview cuts as they will be make by the movie player, and verify that the motion appears continuous from one sequence to another.

The editor is also capable of creating links automatically. The auto-linker function of the editor uses the subject and sub-subject fields to determine whether one segment can be linked to another. One segment is linked to another whenever they describe the same portion (or portions) of the same overhaul operation, or when one segment describes a tool or subassembly used by another segment. In addition, the segment being linked to must be lower in the segment heirarchy than

the segment being linked from, except for the special case of a subassembly. Whenever the content of the segments match, and a link does not yet exist, the editor will create a link. Existing links are not replaced, so that automatic linking may be done after new segments are entered into the database, without replacing those links whose cut points have been optimally adjusted. The auto-linker uses the percentage completion fields in the subject and sub-subject descriptor records to calculate the link exit and reentry positions, such that the link overlaps the correct portion of the segment being linked from. Although one must generally adjust the link positions manually after previewing the cuts, the automatic linker does find all the segments which can be linked together, and roughly positions them.. This is especially usefull for linking commonly used segments such as the tool description shots, or for any overhaul operation which has a large number of segments.

The auto-linker starts at a specified segment. It creates all the possible links for the starting segment, and then does the same for each segment which was linked to the starting segment. It then creates links for each segment which is linked by them, and so on. Thus the auto-linker will create a tree which describes an entire operation. The database creation process consists of defining all the segments by means of segment, subject, sub-subject, and knowledge decriptor records. Then the auto-linker is run to structure the segment

tree by means of the link descriptor records. Then the links are adjusted manually, and the database is ready to be used by the movie player.

C. The Movie Player

The movie player is the program which actually plays personalized movies. It interacts with the viewer and constucts the viewer model. The movie maker uses the viewer model and the videodisc database to dtermine what segments should make up each movie. Movies are played using two videodisc players under control of the movie maker program. Each videodisc player has a copy of the bicycle disc described above, so all sequences can be played from either disc. The following description of the movie maker first has an overview of how movies are generated using the database, and then an explanation of how movies are personalized to the viewer.

The movie player uses most of the data base created by the editor, as well as some files of its own. In particular the movie player does not currently use the subject or sub-subject descriptor records, since it is assumed that the automatic linker has used the subject data to create trees which describe each subject. All of the segments relevent to a subject can thus be accessed via the tree, instead of the subject records. The movie player also has a file containing a profile of each person who has interacted with it; these profiles are updated whenever a movie is created. Although one of the first things done by the movie player is to ask for the name of the viewer, it is best to first discuss the mechanism for expanding the database tree and creating a movie.

Figure 10 is a flowchart of this process.

The movie player first asks for the topic of the movie to be created. This is currently done by displaying a list of possible topics and asking the viewer to enter the topic number. Although this method is somewhat clumsy, the end result is the same as if the viewer could point to a picture of the bicycle and indicate which part is to be overhauled. Each topic is associated with a start node within the database tree. This node, or segment, is the master segment for the topic. It is generally a long shot of the entire overhaul operation. This node is used as a start node for expansion of the tree.

The expansion of the tree is done as follows: The first node is expanded into what are called movie sequences. A sequence is an expansion of a segment descriptor record (see Figure 11). A sequence descriptor record contains all of the segment descriptor fields, as well as additional data describing how a linked segment is to interact with the sequence, and cinematic data such as the speed the sequence is to be played at, and audio/visual interaction. Initially, one sequence is created for each knowledge descriptor record of the segment being expanded. For a long segment there may be twenty or thirty sequences generated; for a short segment there may be only one. In fact, due to the matching of the viewers knowledge with the knowledge description record, some or all of the knowledge portions may be discarded. However, for now, the

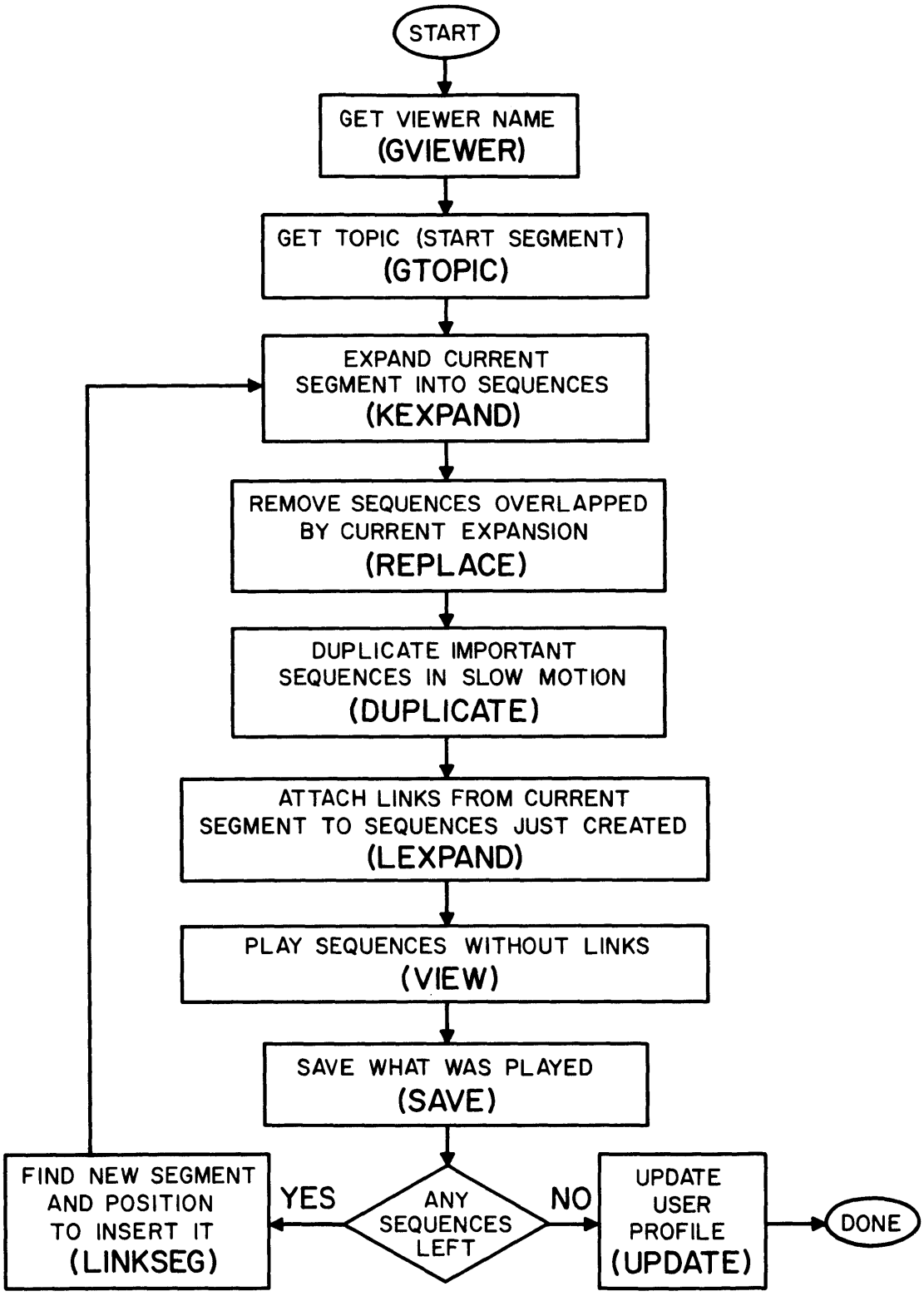


FIGURE 10 THE MOVIE PLAYER FLOWCHART

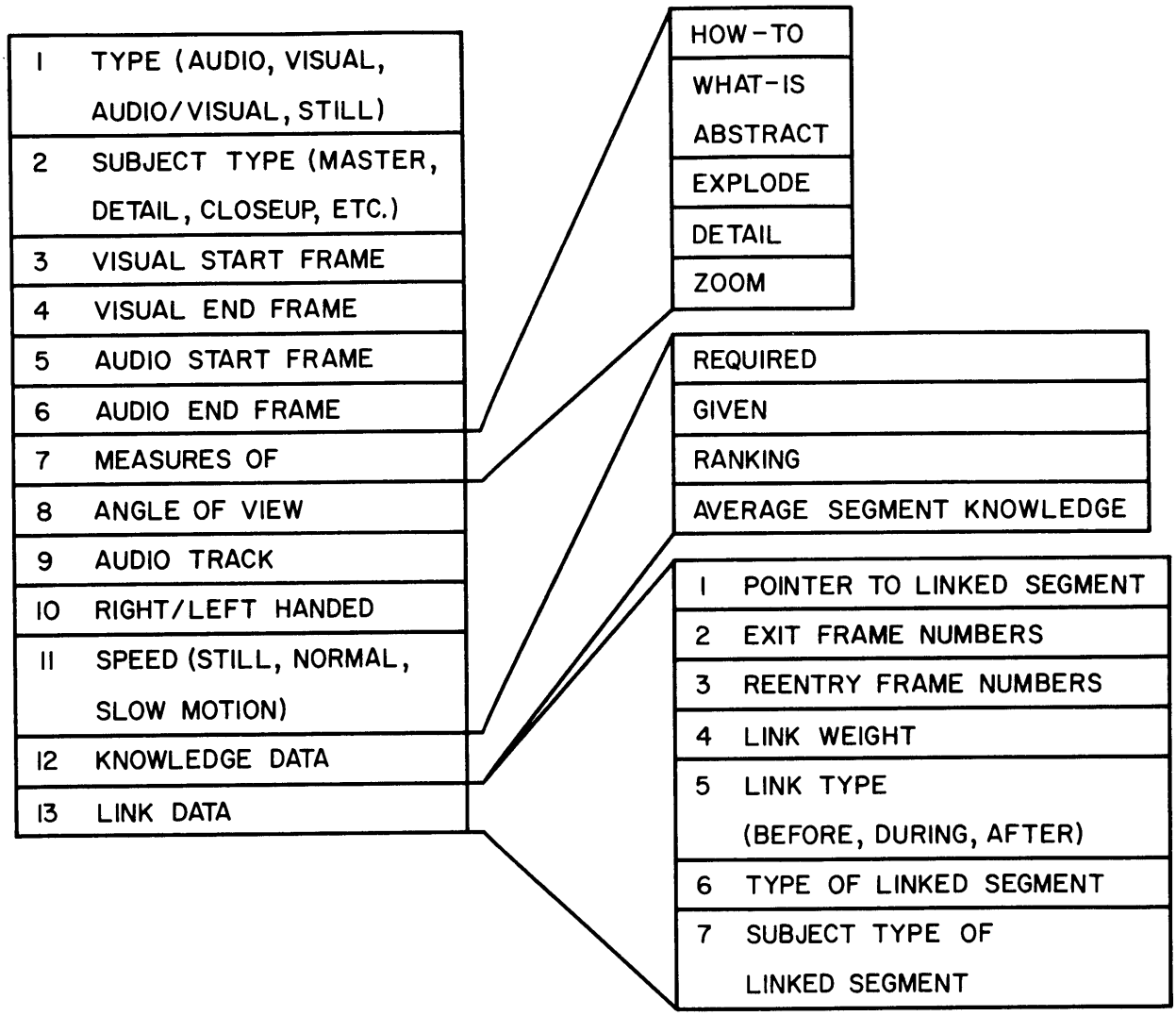


FIGURE 11 SEQUENCE DESCRIPTION RECORD

assumption will be that all expansions are used.

After the knowledge records for a segment have been processed, the links for the segment must also be included. Each sequence can also specify one link. Therefore, each link is stored in the first sequence which overlaps the link exit position. If more than one link overlaps a given sequence, then dummy sequences are created to hold subsequent sets of link information. Once again, various tests are made to determine whether a particular link should be included, and thus become part of the movie.

After all the links for the current segment have been processed, the movie player attemps to play sequences which do not have any links to be expanded. Starting with the first sequence, the movie player will play each sequence in turn on the videodisc until a sequence containing a link is found. Sequences without links are sequences which cannot be expanded any further since there is nothing below them in the tree. Therefore, once sequences have been played they are deleted. On the other hand, sequences still containing links must be expanded further (see Figure 12). If a master segment has just been expanded, then it is likely that all sequences will still contain links and nothing can yet be played. In any case the first remaining sequence becomes the current sequence, and the movie maker uses its link information to find the next segment to be linked.

The segment being linked may go before, after, or in the

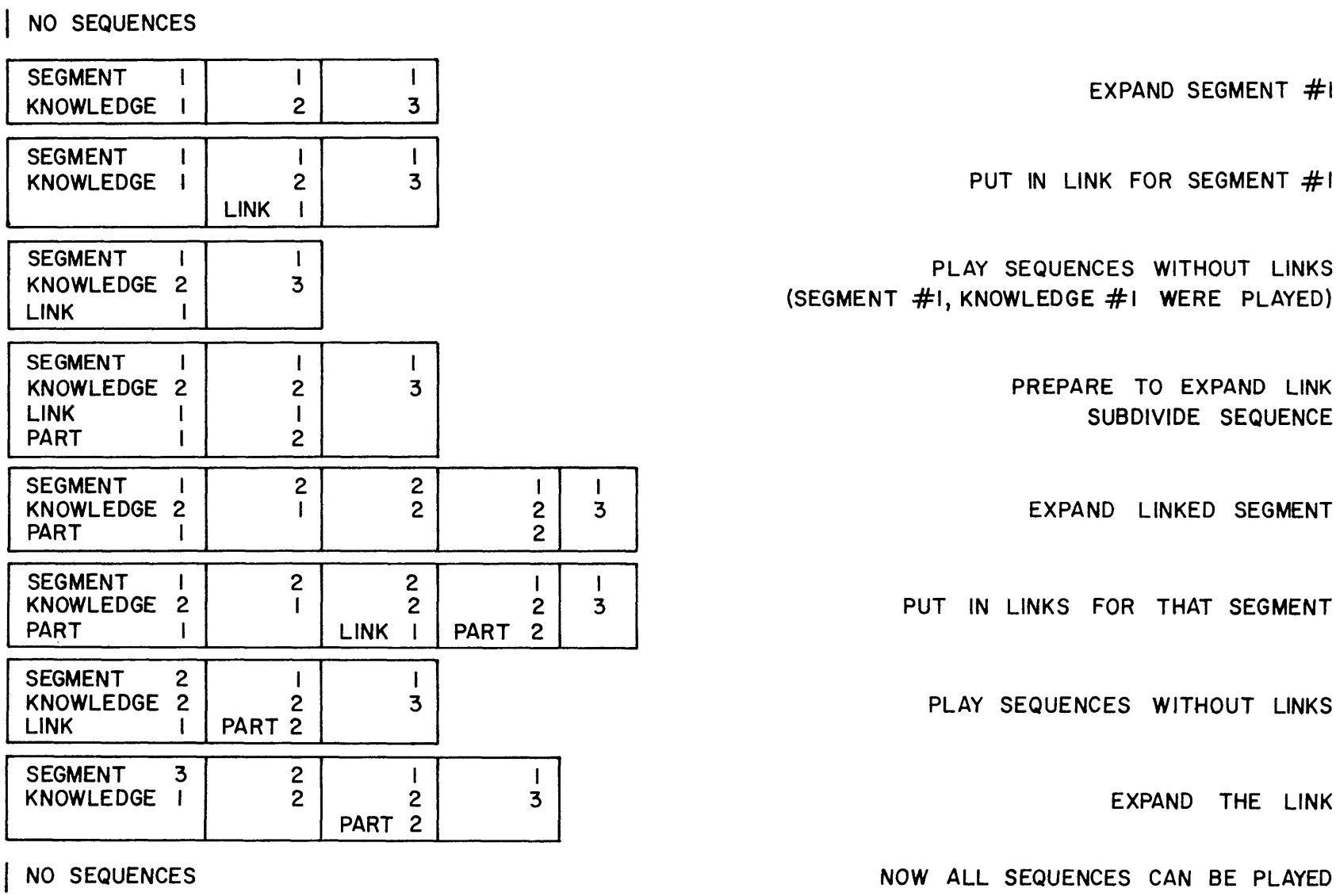


FIGURE 12 THE SEQUENCE LIST

middle of the first sequence, and thus may require splitting the sequence in two. In any case, a pointer is set to where the new sequences derived from the linked segment should be inserted. The new segment is expanded into sequences in the same way as the master sequence described above. These sequences are inserted into the sequence list at the appropriate place. After the segment has been expanded, the succeeding sequences must be checked to see if they will be overlapped by the newly inserted sequences. They may have to be deleted, or shortened. Then the movie player once again plays sequences which do not contain unexpanded links, and expands the first link found. This process continues until all the appropriate segments have been expanded, and all the resulting sequences have been played. That is the way in which movies are generated.

As mentioned above, all expansions of the tree are subject to various tests, thus pruning the tree to fit each viewer's preference. Some of these tests are made to determine whether a viewer is likely to wish (or need) to see a given sequence, some tests are used to compare the value of one sequence with another, and other tests evaluate the cinematic problems of intercutting the various audio and visual components. In order for a sequence to be played it has to pass all of these tests.

Initially, when a segment is expanded into sequences, the knowledge levels of the viewer are compared with the knowledge

levels in each knowledge descriptor record. The viewer has a knowledge level for each segment, as well as each overhaul operation, and a preference for the type of learning ("how-to" or "what-is"). Weights are calculated based on whether the viewer has sufficient knowledge to watch the segment portion, how much will be learned from it, and how closely the mode of teaching matches the preferred learning mode of the viewer. This weight becomes the knowledge ranking of the sequence. In order for a sequence to be created, the knowledge ranking for a portion must exceed a certain value. This value differs depending on whether the segment is a master segment (their portions are almost always used) or another subject type. When the current knowledge portion would overlap another sequence, their knowledge rankings are compared to see which sequence should be used.

When processing the links for a segment another series of tests are made. The first test evaluates whether the segment description corresponds to the viewer preferences: The segment descriptors, such as level of detail, mode of teaching, and measure of abstraction, are matched with the viewer preferences to calculate a weight for the segment. Audio and visual components are also matched, and may even be removed if the viewer has a strong preference for one or the other. Once again, the weight must exceed a certain minimum in order for the segment to be included. If the weight is sufficiently high, then other tests are made: Existing sequences are

scanned to see if they contain links to overlapping segments, thus implying that these other segments describe the same material, most likely in a different manner. If a duplicate segment is found, the weights of the two segments are compared. If one segment has a considerably higher weight it is used, and the other segment is discarded. If both segments have roughly the same weight, the viewer profile is scanned to see which segment has been shown to the viewer most recently; that segment is discarded and the other segment is used. This test is designed to introduce as much variety as possible when equivalent material is available.

The next test, which is made when processing the links for a segment, is to evaluate how each linked segment can be intercut with the sequence from which it will be linked. The cinematic considerations of intercutting audio and visual components are used in this test. For example, two segments with narration can only be played sequentially, since a narration cannot be interrupted in midsentence. On the other hand a picture may be interrupted by another picture, or may be overlapped with a narrative segment. A segment with both audio and visual components may have the picture temporarily replaced by a closeup while the narration continues. There are many possibilities, and the decision involves not only whether the combination is possible, but how to resolve impossible combinations, such as two narrations in parallel (one is generally discarded). In addition, some combinations need to be

played on two videodiscs concurrently; hence they are not allowed if there is only one videodisc on the system. The cutting of picture segments is also subject to some basic cinematic checks to avoid jump cuts. For example, two long shots from similar angles cannot be played in succession, since it would be visually disturbing.

After examining the cinematic interaction of each segment with the appropriate sequence, their relationship is determined to be one of the following. Firstly, the segment may be discarded. It may be placed at the begining or the end of the sequence being linked from. The segment may interrupt the sequence being linked from, or it may be played concurrently with the sequence being linked from. Lastly, if the sequence being linked from is a dummy, the segment may be included without any effect at all. In fact, dummy sequences are used to avoid jump cuts. When a long segment is divided up into many portions, some of these portions will probably not become sequences, due to a low knowldege ranking. In order to avoid jump cuts, dummy sequences are created between the disjoint portions, and a special effort is made to link another segment into these dummy sequences. Still diagrams are used to bridge the jump cuts if no other segment is found to be appropriate.

After passing through all these checks, the segment is almost ready to be included. However, first its weight is compared with the link weight of any sequences which it overlaps. Once again, if the segment link weight is considerably

lower than that of the sequence being overlapped, the segment is not included. This last check prevents the tree from being fully expanded when the viewer would actually prefer the simple explanation provided by the master segment for an operation. Thus by the time a sequence is actually played, it has been checked both against absolute criteria, and against other similar sequences, to verify that the sequence is appropriate for the particular viewer.

Another feature of the movie player is that it will automatically replay valuable sequences in slow motion. If a sequence has a particularly high knowledge ranking, and it is a short closeup, it will be duplicated by the player, and thus be played twice, once at normal speed, and then once again in slow motion. The movie player also saves the last few sequences shown so that the viewer can request to have them replayed at any time.

So far this discussion has alluded to the viewers preferences, without actually explaining how they are kept and updated. Each viewer has a profile record, which is used in all of the checks and decisions described above (see Figure 13). When a viewer enters their name, the player looks the name up in the viewer file. If the name exits, the existing profile is used. Otherwise, a new viewer record is added to the viewer file, and the movie player asks a few questions to aid in initialization of the profile record. The quesions are whether the viwer is right or left handed, and how much the

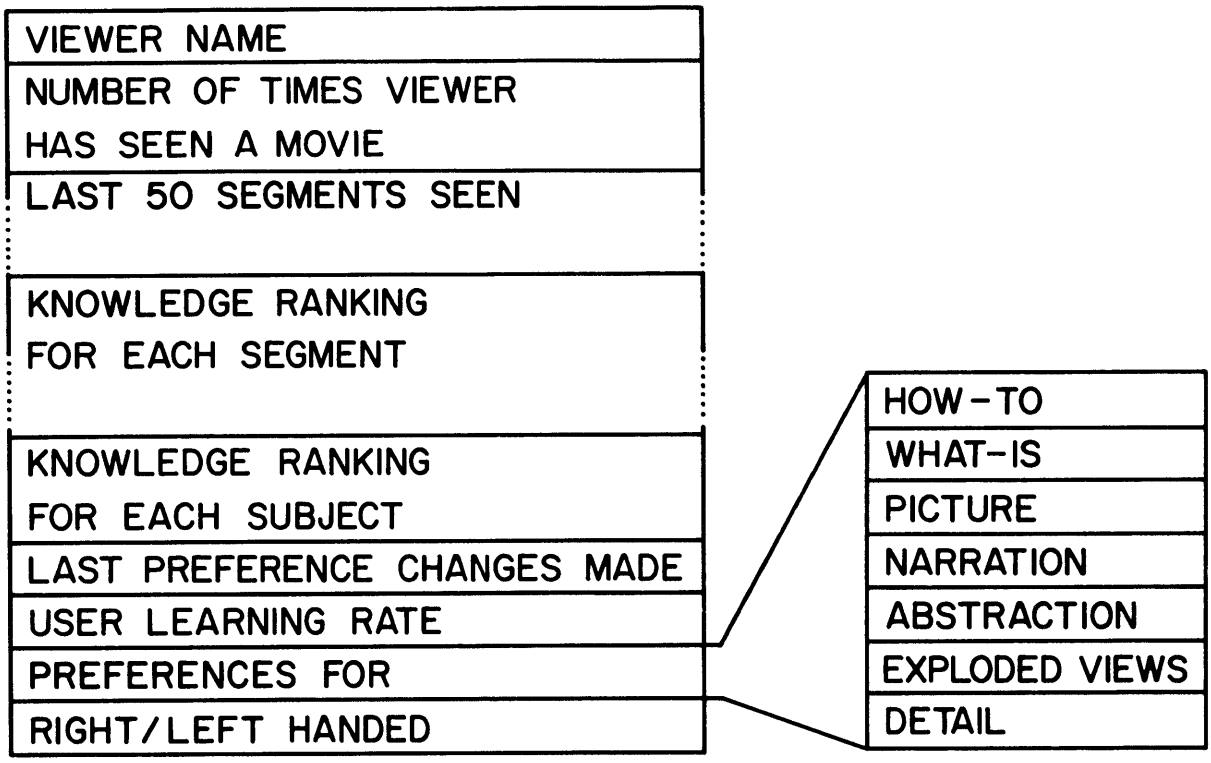


FIGURE 13 VIEWER PROFILE RECORD

viewer know about bicycles, bicycle repair, and mechanical repair in general. These answers are used in the viewers profile, together with neutral preferences for things such as narration, stills, amount of detail, and exploded views. During a movie, the viewer can interrupt the player at any time, and indicate that he or she wishes to alter one or more of their preferences. The player prompts for the type of change, and then alters the viewer profile appropriately. If a viewer makes the same preference request twice in a row, then their profile is alterred more drastically. Thus incorrect assumptions about the viewer can be corrected quite quickly.

Each viewer profile also contains a list of the last 50 segments viewed, thus providing the player with the ability to vary the movie. The player will not show a segment a second time if an equivalent segment is available. Viewer profiles also contain a measure of the viewers knowledge for each topic, and also for each segment in the database. Thus the knowledge ranking reflects both the viewers overall knowledge about a subject, and specific knowledge about a given technique. The appropriate segment and subject values are updated by the viewers learing rate whenever a movie is finished. The viewer's knowledge for all subjects and segments are set to a certain minimum, which is a fraction of the viewers average knowledge for all subjects and segments. Thus a viewer who is familiar with one part of the bicycle is assumed to be some-

what familiar with the techniques of bicycle repair, and is not treated as a complete novice when a new subject is shown.

There are only a few facets of the movie player which have not yet been discussed. One of these is that the player asks how many videodiscs are on the system. This number is used by the player to decide whether certain audio/visual combinations can be played. If there are two videodiscs (the current maximum), the player switches between them, searching on one videodisc while the other is playing. The player can also run without any videodiscs, or can pause before playing each sequence. This is very usefull for debugging purposes. The player can also save the movies which are created. However, this is really a part of the story teller.

D. Story Teller

The story teller is the last component of the movie maker. The story teller is a diagnostic aid which allows one to save movies, replay them, and describe them in terms of the original database. The movie player can optionally create a file which contains all of the sequences shown during a movie. This file can also be played back by the movie player, since it is exactly the same as the list of sequences generated by expanding the tree. The story teller is used to close the loop between the player and the editor. It reads the file of sequences saved by the movie player, and describes each sequence. This is done by displaying the segment, subject, sub-subject, and knowledge descriptor records which describe the sequence. Of particular interest are the subject and sub-subject descriptor records since they describe the content of the movie in English. When these records are displayed, the format is similar to that used by the editor. However, videodisc frame numbers and pointer fields are not displayed, thus making the display shorter, and easier to read. Record numbers are included, to that one can go back to the editor to make any required changes. The story teller actually has much in common with the editor, and uses some of its subroutines.

CHAPTER 5

Program Documentation

A. Conventions

This chapter describes each program and subroutine in the movie maker, and conventions used in accessing the database. It is intended as a document for those involved in follow on work or maintenance. It will first focus on general conventions which apply to the database, and then on subroutines within each program. Appendix A containing complete program listings should also be referenced.

The database consists of a number of files containing fixed length records. Each such file has a unique record format which is defined using a PL1 structure. For ease in typing, each structure has a one letter name as follows:

s - segment descriptor record

u - subject descriptor record

j - sub-subject descriptor record

k - knowledge descriptor record

l - link descriptor record

q - sequence descriptor record

v - viewer profile record

The proper letter is present in each variable used to reference a particular type of record. For example the record count variables are SS, UU, JJ, KK, LL, and VV. Each file is defined as an array of the appropriate structure. A count of the number of records allocated is the first location in each file. This is followed by the records themselves. The prefix "X" is used to reference a record within the file. For example XS(SS).TYPE is used to reference the type field of a segment decriptor record, and XQ(QQ).TYPE the type field of a sequence record. To facilitate the examination of fields within the same record, each structure defining the record type is defined as a based variable. The structures are based on the pointers SSP, QQP etc. Thus once the pointer SSP has been set to point to a segment record, S.TYPE is sufficient to reference its type field. Similarly, Q.TYPE references the type of a sequence record pointed to by QQP. This convention is used throughout, and makes specifications both short and understandable. The structures defining each record type are "in-

clude" files, which are listed in the appendix. In addition there are "include" files which define the entire files for each record type.

The files which make up the database are in the directory >U>RJM>BIKE. Their names are SEGMENT.DATA, SUBJECT.DATA, and so on. There is also an "exec" file, SAVE.EC, which copies all of the data files into corresponding files named SEGMENT.SAVE, SUBJECT.SAVE and so on. RESTORE.EC will copy the files back. There are four sub-directories. >U>RJM>BIKE>EDIT contains all the modules which make up the editor. >U>RJM>BIKE>PLAY contains the movie player, and >U>RJM>BIKE>TELL has the story teller. >U>RJM>BIKE>GOOD-VERSION is used to keep a working version of each program while development work is being done.

B. Editor

The main program of the editor is called EDITOR, and it contains entries for all of the edit commands. Since most of the edit commands need to access more than one record of a particular file, it is difficult to create subroutines which don't have to be passed pointers to all of the files and all the record counts. Thus the editor main program is fairly massive, and actually would be more so, except that the compiler can't compiler anything that big. Instead, some fairly major sections have been made into subroutines, which require many arguments. More traditional functions such as display routines, and command input are also subroutines.

The editor tries wherever possible to reference fields independently of which record they are in. This reduces the lines of code, and makes changing the editor much easier since changes only need be made in one place. The editor has a current mode which defines the type of record currently being accessed. Modes are:

Ø Subject record

1 Segment record

2 Sub-subject record

3 Knowledge record

4 Link record

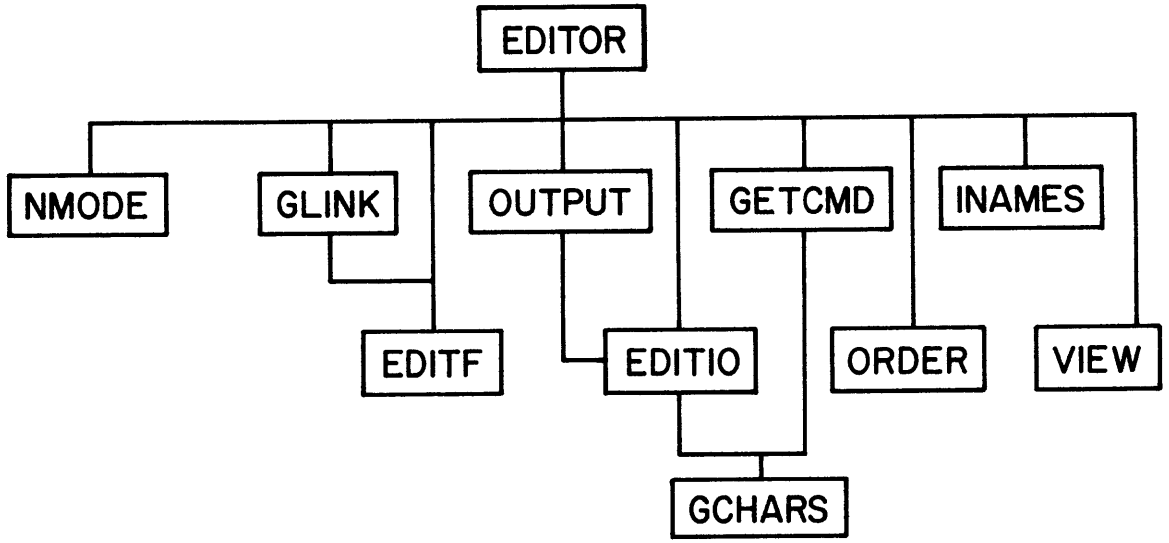


FIGURE 14 EDITOR SUBROUTINE FLOW

For each record type there are variables decribing how many fields the record contains, how long each field is, what type of value it contains, and what the name of the field is when it is displayed on the screen. There are also pointers and record counts for each record type. These variables can be referenced either by name or as a function of the current mode. For example UU, the number of the current subject record, can also be referenced as XX(Ø), and SS is equivalent to XX(1). These variables are defined in the EDIT and NAMES "include" file. Upon program execution, the names and types of each field in each record type are initialized by the INAMES subroutine. This subroutine is edited to initialize new types or names when the usage of a field changes.

The editor main program has entries for each command. Simple commands, such as moving to the next record or the next field within a record, are in EDITOR since it has pointers to all the database files. These commands are done independently of mode whenever possible although there are always execptions. Similarly, commands which change the mode are also done in EDITOR, although a small subroutine is called to do checking for each type of mode change. This subroutine, called NMODE (for new mode), simply removes some code from EDITOR. The Add, Copy, and Delete commands call entry points in the EDITF module. Pointers to all the files are passed, and EDITF also uses the file definition "include" files. Adding and deleting are not very difficult, but the updating of po-

inters and counts takes up a fair bit of space, and must be done separately for each record type. Segment and subject record types are always added and deleted as a unit. The Copy command is just a special case of Add.

The module EDITIO contains routines to display an entire record. There are separate entry points for segment, sub-subject, knowledge, and link descriptor records. Subject descriptor records are displayed together with their associated segment descriptor record. The routines within EDITIO call a subroutine called GCHARS to obtain the contents of a specfic field. GCHARS returns not only the value of the field, but also the field name (that set up by INAMES), and the meaning of the value. The EDITIO routines loop through the specified record, displaying the name and contents of each field. The meaning of each field is stored in the dictionary which was described in the previous chapter. This dictionary is created and accessed by GCHARS.

Input requests for the editor are done by the GETCMD routine. Since the editor always opens a field within a record, GETCMD first calls GCHARS to get the value of the current field. It displays the contents of the field, and then prompts for a new value. GETCMD uses the type paramater of the field to prompt for a specific type of input. It always looks first for numerical input. If an input conversion error occurs, GETCMD then uses the length of the character string to distinguish between commands and character strings. GETCMD

calls the REPLACE entry within GCHARS to enter a new value, or character string. REPLACE updates values in the descriptor records, as well as the dictionary; REPLACE prints the dictionary entries, and inserts new entries, as requested.

The VIEW module runs the videodisc for the View and Watch commands. The View command plays the videodisk sequence specified by the current descriptor record, and previews cuts for links. In order to cut between a segment and a segment which is linked to it, VIEW must also access many records within the database. Therefore VIEW is also passed pointers to the entire database. VIEW searches the knowledge description records for a segment to find a reasonable starting position which is not in the middle of a sentence, but does not result in the playing of long sequences before cuts. If there are two videodiscs then VIEW cues up both videodiscs before playing either one, thus enabling it to cut from one segment to another instantly. If there is only one videodisc then VIEW searches for and plays each segment in turn.

The ORDER module puts all the sub-subject, knowledge, and link entries for a particular segment in order of their frame numbers. This may be done while editing; it is always done by the movie player, to which the order is crucial. The OUTPUT module displays the entire database. It simply steps through each segment, calling EDITIO to display first the segment descriptor record, and then each sub-subject, knowledge, and link descriptor record.

The GLINK module is the most complicated part of the editor. It is the module which creates links automatically. GLINK is a series of nested DO loops which scan the database, comparing the current segment and its subject and sub-subject records, with those of all the other segments in the database. GLINK firstly checks all other segments to see whether they are subassemblies or tool usage descriptions used by the current segment. This is done by matching the subject names and types. GLINK also looks for overviews and introductions for the current subject. GLINK then looks for other segments which describe the same bicycle component as the current segment, and which do not conflict with the current segment regarding assembly/disassembly or the front/back of the bicycle. For each match found, GLINK calls an internal routine called MLINK (maybe link) with an argument specifying the type of the link (subassembly, tool description, closeup etc.). MLINK first of all makes sure that no such link already exits. This is so auto-linking can be run again, without destroying previous links which have already been manually adjusted. MLINK then creates the link by calling the same Add subroutine used by the Add command. MLINK calcuates he exit and reentry position of the link using the starting and ending percentage fields from the current sub-subject description record and the subject description record of the newly linked segment. GLINK attempts to snap link points to the start or end positions of knowledge portions so that links don't occur in the middle of

sentences if possible. Most links have to be manually adjusted in any case.

All of these modules are bound together to form the EDIT program which is the editor. The Bind file is called EDIT.BIND. Other command files of interest are GLIST.EC and LIST.EC which respectively generate and print listing files.

C. Movie Player

The movie player lends itself to structuring much more easily than does the editor. The main program is called PLAYER It calls major subroutines which correspond to steps in the process of expanding the database tree (see figure 10). Each subroutine is fairly well self contained, and a minimum of arguments are passed back and forth. A global common area is used by all the subroutines. This common area contains generally usefull variables such as the pointer to the current viewer, the number of sequences, a list of the segments shown so far, and a description of the segment currently being expanded. This common area is defined as a static external structure by an include file called PLAY which is used by all subroutine modules. PLAY also defines some generally usefull variables, such as the current sequence number, which are reallocated in each module. The basic functions of each subroutine are described below; specific tests and decisions are documented in the code.

The SEQSUB module contains subroutines for initializing the sequence list and inserting, copying, and deleting sequences. Sequences are allocated in the same way as any other type of record; the first location in the file is a count of the total number of sequences allocated, and space is reclaimed by setting the type to zero on a delete. Since sequences have to be inserted and deleted frequently, the actual order

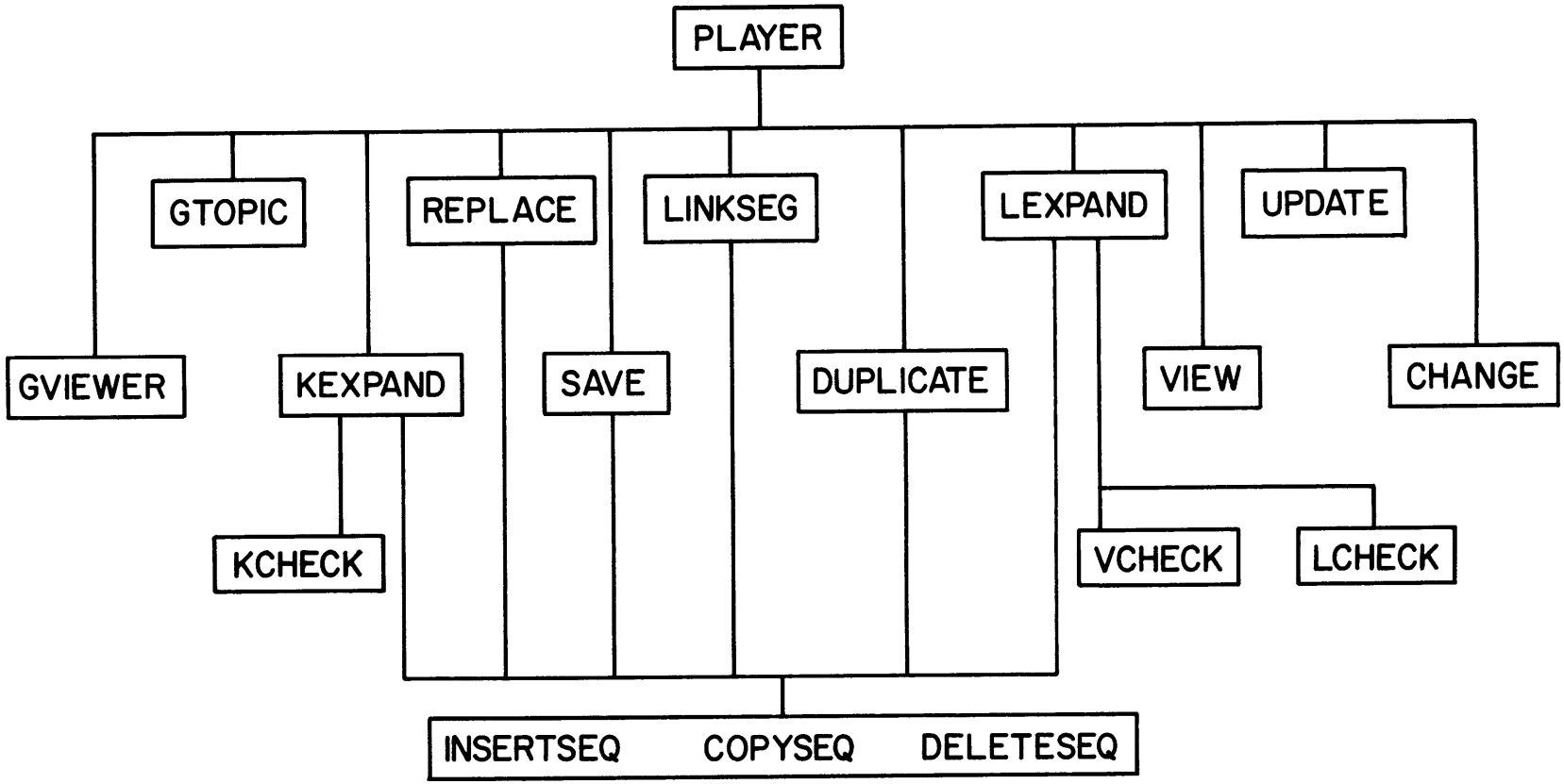


FIGURE 15 MOVIE PLAYER SUBROUTINE FLOW

of sequences is kept in a pointer array called QPTR. The routines in SEQSUB update this array and makes sure that there are no empty entries in the pointer array. The array makes it easy to insert or delete a sequence; one entry in the sequence file is changed and then pointers in the QPTR array are shuffled to reflect the new order. QPTR is part of the global common area.

The first subroutine called by PLAYER is GVIEWER (get viewer) which asks the viewer for his or her name. If the name exits in the viewer file the global pointer VVP is set to point to the viewer profile record. If the name is not known, a new profile record is created, and the new name is inserted. A few questions are then asked to obtain values used in initializing the viewer profile, and then VVP is set to point to the new viewer profile. The VIEWER subroutine also contains code to ask for values for each field in the viewer profile record. This is primarily a debugging aid which allows the creation of "model" viewers. Two GOTO's are used to either execute or skip over this code; they may be included or commented out as required.

After the viewer has been chosen, GTOPIC (get topic) is called to find the start node for the movie. Currently GTOPIC just asks for a topic number (1-8) and has a assembled value of the start node for each topic. In the future, GTOPIC should be extended to use a picture of a bicycle, and a touch sensitive display.

After the topic has been chosen, the main loop through the expansion process is started. The node being expanded is always in the variable SS in PLAYER. Because PLAYER is the only module in the movie player with pointers to all the files, it scans the knowledge file to accumulate a list of pointers to all the knowledge descriptor records for the current segment. PLAYER then calls KEXPAND (knowledge expand) to expand out the knowledge portions. KEXPAND goes through each knowledge record and calls KCHECK (knowledge check) to calulate the knowledge ranking for each record. KCHECK returns both the ranking and a yes/no response as to whether the portion should be used. If it is to be used, KEXPAND first scans other sequences from the current segment to see if the portion is part of an audio/visual combination which has different knowledge records to reflect varying amounts of knowledge. If this is true the the new knowledge portion is combined with the existing sequence containing its other half. Otherwise a new sequence is created. KEXPAND also tests for the case of two segments being played concurrently. Then the segment which was linked to is called a "baby" of the sequence linked from. When expanding a "baby" KEXPAND makes sure that the sequence which contained the link now also contains a pointer to the sequence which describes the "baby". This is so the VIEW subroutine can play the two sequences concurrently. If jump cuts are likely to occur because some knowledge portions are not used, then KEXPAND creates a dummy seqeunce. Dummy se-

quences have links inserted into them by LEXPAND, as described below. KEXPAND also keeps track of which segments and subjects have been successfully expanded, so that the viewer's knowledge for them can be updated at the end of the movie. It should also be noted that some of the segment's attributes, such as its type, are not read from the segment record, but from a globl common location. This is because in the process of linking, a segment's attributes may have been changed; for example, a segment containing both audio and visual components may be treated as a segment having visuals only.

After a segment has been expanded into sequences by KEXPAND, then PLAYER calls REPLACE to update any segments which may have been overlapped. REPLACE looks for sequences which are from the same node as the one just expanded. If the segment just being expanded is the start node for the topic, then nothing can be overlapped. REPLACE uses the original link exit and entry points to determine whether a sequence should end prematurely, start later, be removed altogher, or need not be affected. Because audio cannot be interrupted, audio visual segments which are shortened have their audio removed.

After the REPLACE subroutine call, PLAYER then scans the link description records for the segment being expanded, and creates a list of link pointers. This is just like the list of knowledge pointers prepared for KEXPAND. PLAYER also prepares a corresponding list of pointers to each of the segments

being linked. This once again is because the subroutines don't have pointers to the entire files. If a segment has no links, PLAYER assumes that this segment might be rather quick and will attempt to show it twice. PLAYER calls DUPLICATE, which checks to make sure that the sequence got a high knowledge ranking, and that the segment is not long enough to be tedious if seen again. If the segment passes these tests, then DUPLICATE creates a dupliciate of the sequence immediately following the original. The new sequence is set to run at half speed. If the sequence is very short, then a portion of the sequences before and after the current one may also be duplicated to make the replay more continuous.

If there are any links, LEXPAND (link expand) is called to process them. LEXPAND looks through the sequences just created by KEXPAND to find the first sequence which overlaps the exit point of the link. This sequence is the one which will contain the link to the new segment. LEXPAND first calls VCHECK to see if the viewer is likely to wish to see the segment being linked. Like KCHECK, VCHECK returns a weight and an answer. LEXPAND can influence the decsision of VCHECK by its initial setting of the weight. LEXPAND sets the initial weight to ensure that the dummy sequences introduced by KEXPAND have link information inserted, thus removing possible jump cuts. VCHECK calculates a weight based on the matching of viewer preferences and the segment in question. It then sees if any other sequence contains link information to a seg-

ment with the same information. If so, VCHECK decides which segment is to be seen, and may remove the link information from an existing sequence.

If VCHECK says that the current segment is acceptable, then LEXPAND calls LCHECK (link check) which looks at the types of the segments and decides if, and how, they can be played (sequentially, concurrently, or one interrupting the other). LCHECK returns the type of visual relation between the two segments. If a link can be made, then the information describing the link is copied into the sequence. If the sequence already has a link, then a dummy sequence is created to contain the link information. When saving link information, the crucial variable is the pointer to the segment being linked. If this pointer is null, then no link exists. Other variables, such as the segment number, are used as an audit trail by the REPLACE subroutine after the link has been expanded, and thus do not get reset. LEXPAND also does some special checking of stills, attempting to fill in any possible gaps within the current segment with still frames.

After all the links for the current segment have been expanded by LEXPAND, then PLAYER tries to play sequences which do not have links. This is done by the VIEW subroutine. The view subroutine simply plays sequences until it finds the first one which has an unexpanded link. It also attempts to combine sequences which are physically continuous on the videodisc. Such sequences are generally from separate knowlege

portions which were part of the same segment. VIEW also skips over dummy sequences which are still present. If there are two videodiscs, then VIEW alternates between them, only waiting for the first videodisc when the second one is cued up and ready to play. VIEW sets the registers in the videodisc controller so that once the first videodisc has stopped playing, only a single command has to be sent to start up the second videodisc. VIEW returns to PLAYER the number of the first sequence which had an unexpanded link.

PLAYER then calls SAVE, which saves what was played and then deletes those sequences from the sequence list. For each sequence shown, SAVE first of all saves the sequence in a push down list of the last 5 sequences played. SAVE then updates the viewer profile list of the last 50 segments seen by the viewer. If a story teller file is being created, then SAVE also stores the played sequences in the ouput file for the storyteller. SAVE uses an "include" file from the story teller to define the file format. Finally, SAVE deletes the sequence to remove it from the current sequence list.

After all the played sequences have been deleted, the first sequence is now the one with an unexpanded link. PLAYER calls LINKSEG to expand the link. LINKSEG looks at the link information in the sequence and decides whether the new segment goes before, after, or the middle of the current sequence. If the linked segment interrupts the current sequence, then LINKSEG splits the current sequence in two.

LINKSEG also saves information from the link block (such as its type) in global common locations used by the routines which subsequently expand the linked segment. LINKSEG returns the segment number of the segment just linked to, and the position in the sequence list where new sequences should now be inserted. The main loop in PLAYER then goes back to expand this new segment into its sequences.

When VIEW has played all the sequence, PLAYER calls UPDATE. UDPATE simply makes the viewer more knowledgeable for the subject seen, and for all the individual segments shown. More than one subject may have been seen if a subassembly was included. UPDATE also decreases the rate as which the viewer learns, thus making knowledge progressively harder to obtain.

The CHANGE subroutine interacts with the viewer to change the viewer profile. Whenever the viewer types a break character, a flag is set, and at the next convenient point, the CHANGE subroutine is called. The CHANGE subroutine allows the user to specify changes to one of six preferences types. These changes are immediately reflected in the viewer profile, and thus in any subsequent decisions about which segments to play. The CHANGE routine remembers the last user request for each preference type, and another similar request changes the viewer preference by a larger amount.

All the modules which make up the player are bound together by PLAY.BIND to create the program PLAY. The files GLIST.EC and LIST.EC generate and print listing files respec-

tively. The file COMPILE.EC can be used to recompile all the modules in the player, should changes be made to one of the "include" files.

D. Story Teller

The story teller program is called TELLER. It opens a specified sequence file previously created by the SAVE routine in the movie player. TELLER steps through each sequence and searches all the descriptor records for the segment which is shown by that sequence. All descriptor records which overlap the sequence are displayed. The story teller uses a module called TELLIO to actually display the records. TELLIO is very similar to EDITIO; the two modules have the same entry points but TELLIO does not display videodisc frame numbers or pointer fields. The story teller uses the INAMES and GCHARS routines from the editor so that all field names and dictionary meanings are consistent. The format of the saved file is defined by an include file. The file contains the name of the viewer it was prepared for, the number of sequences, and all the sequences shown during the movie. The bind file TELL.BIND creates the program TELL.

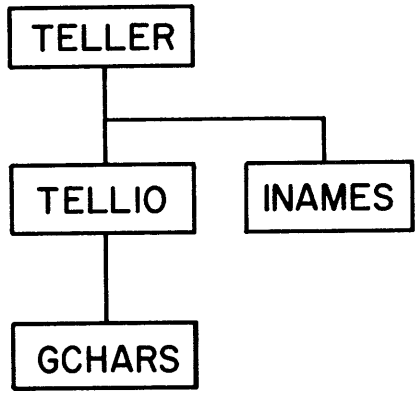


FIGURE 16 STORYTELLER SUBROUTINE FLOW

CHAPTER 6

Continuation Work

There are a number of areas in which the current movie maker does not live up to its initial design. The most glaring of these, at least to the viewer, is that interaction with the movie player is rather clumsy. The viewer should be able to interact with the player via a touch sensitive screen, using menus which include pictures of a bicycle from the videodisc. This would make the system much more accessible.

Another part of the original design which was never implemented is animation. Originally animation and film were to be of equal importance. However, the current system concentrates almost totally on live film excluding other media such as animation, and, to a certain extend, still frames. Therefore, inclusion of other medium is definitely an area for future investigations.

Other areas of improvment are in the internals of the movie player. Currently, the player does not access the subject or sub-subject files, and thus does not really know what it is that is being played; it is somewhat disturbing to have an intelligent program which does not understand its data. Inclusion of the subject data would not only make the player

smarter, but would also permit the implementation of a query system, whereby the viewer could request information on a given subject. Subject information would permit the movie player to make better choices between large numbers of simple segments, such as that needed to intelligently put togther a slide show as opposed to a movie. The inclusion of the subject files, although not in itself difficult, would require a modification of the way files are accessed under MagicSix.

Another area which must be considered is the scheduling and interaction problems created when there are more than two videodiscs in the system. Many of the film making decisions depend on exactly how many videodiscs there are. It is important to know how many different sources there are for audio and visual components, both for cueing and for concurrent playing. The current player does not deal with the possiblities introduced by more than two videodiscs. In addition, the current videodisc scheduler is only capable of alternating between videodiscs. It does not play videodiscs concurrenly, and would need to be expanded for more videodiscs. One reason that the current scheduler is so restricted is that some of the functions necessary for effecient computer control of many videodiscs do not yet exit, and make such scheduling difficult.

The last area of improvement to the current movie maker is in the area of modeling the viewer. The current model is fine as far as it goes, but suffers from a lack of feedback.

Much of this is because the movie player has always operated in an artical environment, where "tyical" viewers have been created. One should let a large number of viewers interact with the player, and then use group preferences as a guide in responding to future viewers. The current player does not attempt to relate one viewer with another.

CHAPTER 7

Looking Ahead

This project was an initial investigation into personalized movies. A number of new avenues have been explored, and are suitable for more research. One area is the creation of movies using a computer. This requires special filming techniques to prepare the material for access by the computer, and computer programs construct a a movie. The problems involved in fitting together various audio and visual components were much more complex than first realized, and should be investigated further. Such an investigation should consider both the editing decisions which are present in traditional film making, and the editing decisions which are introduced when sound and picture become individual components, which can be mixed in many ways. One use for a well developed program to create movies would be to aid in editing film. By continually changing the computer description of the material, many more editing possiblities could be previewed in a smaller amount of time and with much less effort than with present day editing techniques.

The personalized movie system developed as a part of this project did not have a great deal of knowledge about the sub-

ject matter being shown. However, even this small amount of knowledge, coupled with viewer personalization, resulted in a system which appears intelligent. A more rigorous representation of knowledge, and the resultant refinement of the viewer model, should create a more intelligent system which can be used to model more complicated objects. It is conceivable to develop a system which could instruct one in the repair of an automobile.

More advanced maintenance and repair systems can be developed by additional advancement in boththese areas. Such systems could be more active than the personalized movie system described here. They would not model the repair person by waiting for requests, but rather by sensing whether the repair was proceding correctly. This can be done by new technologies which allow the tracking of objects in three dimensions. These personalized systems will not simply create movies and wait for direction. They would not just describe the way in which an operation should be done, but will be able to react to unforeseen problems or failures. They could more accurately be described as coaching systems. With the continued advent of technology such a system could reside in a microprocessor, and eventually become part of a mechanics toolbox.

REFERENCES

Brown, J.S. & Burton R.R. Systematic understanding: Synthesis, analysis, and contingent knowledge in specialized understanding systems. In D. Bobrow and A. Collins (eds), Representation and Understanding: Studies in Cognitive Science, New York: Academic Press, 1975.

Engelbart, D.C. et al. The augmented knowledge workshop. Proceedings of the National Computer Conference, 1973.

Goldstein, I. The computer as coach: An athletic paradigm for intellectual education. Massachusetts Institute of Technology, Artifical Intelligence Laboratory, AI Memo 389, 1976.

Irby, C. Display techniques for interactive text manipulation. Proceedings of the National Computer Conference, 1974.

Mann, W.C. Dialogue based research in man-machine communication. USC Informatin Sciences Institute, November 1975.

Miller M.L. & Goldstein I.P. Planning and Debugging in a computer coach for elementary programming. Massachusetts Institute of Technology, Artifical Intelligence Laboratory, January 1978.

Negroponte, N. The Metaphysics of Television, Proceedings of the Workshop on Methodology of Interaction, Seillac, France, may 7-10, 1979.

Negroponte, N. Fields, C. Using new clues to find data, Proceedings of 3rd International Very Large Data Base Conference, Tokyo, 1977.

Negroponte, N. (B. Gilchrist, Editor) On being creative with computer-aided design, invited paper, Information Processing 77 IFIP, North-Holland Publishing Company, 1977.

Negroponte, N. Idiosyncratic Systems, ONR Report NN-100-1, March 31, 1977.

Negroponte, N. Idiosyncratic systems approach to computer graphics, User-Oriented Design of Interactive Computer Graphics, ACM/ SIGGRAPH Workshop, S. Treu (editor), October 14-15, 1976.

Nelson, T.H. A file structure for the complex, the changing, and the indeterminate, ACM 20th National Conference, 1965.

Robertson, G., Newell, A., Ramakrishna, K. ZOG: A man-machine communication philosophy. Department of Computer Science, Carnegie-Mellon University, 1977.

Snediker, J.M. A self organizing program for describing concepts, ACM 20th National Conference, 1965.

ACKNOWLEDGEMENTS

I would like to thank Andy Lipman for providing incentive, ideas, and much assistance, all of which were necessary for the completion of this project. Institute regulations not withstanding, he was my Thesis Advisor. I would also thank Professor Nicholas Negroponte for his original conception and continued support of the project, and the Office of Naval Research for their sponsorship. Additionally, Marek Zalewski for doing the filming, Peter Fiekowsky for overhauling Andy's bicycle, and Tom Boyle for writing the videodisc software. Lastly, and by no means leastly, Sandy, for her patience and support.