

DEPARTMENT  
DEPARTMENT  
DEPARTMENT  
SCIENCE  
SCIENCE  
SCIENCE  
COMPUTER  
COMPUTER  
COMPUTER

UNIVERSITY OF WATERLOO  
UNIVERSITY OF WATERLOO  
UNIVERSITY OF WATERLOO



*A Computer System for  
Smooth Keyframe Animation*

*D.H.U. Kochanek  
R. Bartels  
K.S. Booth*

*CS-82-42*

*December 1982*

# A Computer System for Smooth Keyframe Animation<sup>1</sup>

*D.H.U. Kochanek<sup>2</sup>*  
*R. Bartels<sup>3</sup>*  
*K.S. Booth<sup>3</sup>*

## Abstract

This thesis describes an interactive keyframe animation system which is currently in use at the National Film Board of Canada. After a description of this production system the problems of linear keyframe interpolation are analyzed, and a new algorithm is presented which significantly improves the quality of computer animation by generating motion along smooth curved paths. A preliminary design for a new system based on this algorithm is discussed, and the problem of temporal aliasing in computer animation is introduced as a subject for further study.

<sup>1</sup> This report was originally submitted as a master's thesis to the Mathematics Faculty of the University of Waterloo by the first author. The research reported here was supported in part by the Natural Sciences and Engineering Research Council of Canada, the National Film Board of Canada, the Province of Ontario under its BILD program, the University of Waterloo and the Computer Graphics Laboratory.

<sup>2</sup> Present Address: French Animation Studio, P-36, National Film Board of Canada, Box 6100, Station "A", Montreal, Quebec H3C 3H5. (514) 333-3434.

<sup>3</sup> Present Address: Computer Graphics Laboratory, University of Waterloo, Ontario, Canada N2L 3G1. (519) 886-1351.

## Table of Contents

1. INTRODUCTION .....	1
1.1. Cel Animation .....	1
1.1.1. Inbetweening .....	2
1.1.2. Inking, Opaqing, and Painting .....	5
1.1.3. Merging and Filming .....	5
1.2. 3-D Animation .....	6
2. THE "CINANIM" KEYFRAME ANIMATION SYSTEM .....	7
2.1. Historical Background .....	7
2.2. The User's View .....	8
2.2.1. Operating Environment .....	8
2.2.2. System Description .....	10
2.2.2.1. DRAW .....	13
2.2.2.2. MANIPULATE .....	16
2.2.2.3. GENERATE .....	18
2.2.2.4. SKELETON .....	19
2.2.2.5. COMPOSE .....	20
2.2.2.6. INTERPOLATE .....	21
2.2.2.7. VIDEO .....	23
2.2.2.8. FILM .....	25
2.2.2.9. UTILITY .....	28
2.3. The Programmer's View of the Interpolation Process .....	29
2.3.1. Problems in Computer Assisted Inbetweening .....	29
2.3.2. Previous Approaches to Computer Assisted Inbetweening .....	34
2.3.3. "Intelligent Inbetweening" .....	39
2.3.3.1. The Human Inbetweener .....	39
2.3.3.2. Desirable Properties for Automated Inbetweening Systems .....	40
2.3.3.3. An Algorithm for Smooth Keyframe Interpolation .....	41
3. WHAT NEXT? .....	45
3.1. Towards the Next Generation .....	45
3.1.1. Proposed Hardware Configuration .....	46

3.1.2. Software Design . . . . .	49
3.1.2.1. Menu Design . . . . .	49
3.1.2.2. DRAW . . . . .	50
3.1.2.3. SCRIPT . . . . .	52
3.1.2.4. INBETWEEN . . . . .	56
3.1.2.5. FILM . . . . .	56
3.1.2.6. UTILITY . . . . .	57
3.2. Temporal Aliasing in Motion Pictures . . . . .	58
3.2.1. Motion Picture Cameras . . . . .	59
3.2.2. Motion Picture Projectors . . . . .	60
3.2.3. Temporal Anti-Aliasing . . . . .	62
3.2.4. Solution Approaches in Computer Animation . . . . .	64
Appendix A: Glossary . . . . .	67
Appendix B: Bibliography . . . . .	73
Appendix C: Source Listings for Interpolation Procedures . . . . .	76
Appendix D: Demonstration Film . . . . .	81

## List of Illustrations

1.1.	A sequence of inbetweens based on three keys . . . . .	3
1.2.	Animator entering drawings at workstation . . . . .	4
2.1.	Animation system hardware configuration . . . . .	9
2.2.	Display format for alphanumeric terminal . . . . .	10
2.3.	Digitizing tablet with menu overlay . . . . .	11
2.4.	Overview of steps required to produce an animated film . . . . .	12
2.5.	A picture consisting of three cels . . . . .	14
2.6.	The importance of correct stroke ordering . . . . .	15
2.7.	The DRAW menu . . . . .	15
2.8.	The MANIPULATE menu . . . . .	17
2.9.	Picture manipulation wheels . . . . .	17
2.10.	The GENERATE menu . . . . .	19
2.11.	The SKELETON menu . . . . .	20
2.12.	The COMPOSE menu . . . . .	21
2.13.	The INTERPOLATE menu . . . . .	22
2.14.	Available output types . . . . .	22
2.15.	The VIDEO menu . . . . .	24
2.16.	The FILM menu . . . . .	26
2.17.	Camera with bi-pack magazine . . . . .	27
2.18.	The UTILITY menu . . . . .	28
2.19.	Linear interpolation between two keys . . . . .	30
2.20.	Discontinuities in the direction of motion . . . . .	31
2.21.	Discontinuities in the speed of motion . . . . .	32
2.22.	Distortion caused by rotation being simulated through linear interpolation . . . . .	33
2.23.	Movement in perspective . . . . .	34
2.24.	Motion Path ("P-curve") for a bouncing ball . . . . .	35
2.25.	Skeleton cel and corresponding detail cel . . . . .	36
2.26.	Overlapping actions . . . . .	38
2.27.	Moving point constraints . . . . .	38
2.28.	Basis functions for Hermite interpolation . . . . .	43

3.1.	Proposed hardware configuration . . . . .	47
3.2.	Proposed animator's workstation . . . . .	48
3.3.	File structure for animation data . . . . .	51
3.4.	Sequence dope sheet . . . . .	53
3.5.	Composition dope sheet . . . . .	54
3.6.	Hierarchy of transformations . . . . .	55
3.7.	Burn-through on colour negative . . . . .	57
3.8.	The "wagon wheel phenomenon" . . . . .	59
3.9.	Light input function for a 180° rotary camera shutter . . . . .	60
3.10.	Light output function for a two blade projector shutter . . . . .	61
3.11.	Blurring of motion in live action and animation . . . . .	64

## Preface

This thesis describes the current state of an ongoing research project in computer animation at the National Film Board of Canada. Chapter 1 introduces some of the major concepts in traditional and computer assisted animation. Section 1.1 provides an overview of the different phases of 2-D cel animation and their potential for automation, while Section 1.2 briefly describes 3-D animation techniques.

Chapter 2 discusses the keyframe animation system currently in use at the National Film Board. Section 2.1 sketches the historical context in which it was developed, and Section 2.2 describes the system in detail. The original design and implementation of this system were not part of this thesis. However, an important part of the work presented here has been the restructuring and redesign of large sections of the animation system software. These changes were necessary to upgrade the system's functional power and to improve the design of its user interface. The system description in Section 2.2.2 reflects these changes. The parts of the system which have undergone the most major revisions are discussed in Section 2.2.2.2, Section 2.2.2.7, and Section 2.2.2.8.

Section 2.3 takes a closer look at computer assisted inbetweening. The problems of linear keyframe interpolation are analyzed, and some previously published approaches to computer assisted inbetweening are described. Section 2.3.3 presents a new algorithm for smooth keyframe animation based on Hermite interpolation. A prototype interpolation process using this algorithm has been implemented as part of the research for this thesis. A test film produced with this prototype demonstrates that the new algorithm represents a significant improvement over straightforward linear interpolation.

Chapter 3 discusses some ideas concerning directions for further research. Section 3.1 describes a preliminary design for the next generation of the animation system while Section 3.2 introduces the concept of temporal aliasing in motion pictures. The problem of temporal aliasing has long been recognized in traditional animation. In computer animation it has only recently become the subject of discussion, and no clear description of the phenomenon has been published. Section 3.2 is intended to provide an introduction to the problem and suggest some directions for further research.

*"Animation is not the art of drawings that move, but the art of movements that are drawn. What happens between each frame is more important than what happens on each frame. Animation, therefore, is the art of manipulating the interstices between each frame."*

*Norman McLaren, National Film Board of Canada*

## 1. INTRODUCTION

Animation and special effects are among the most expensive areas of motion picture and television production. High quality film animation requires the preparation of 24 images, each one differing only slightly from the previous one, for each second of viewing time; standard television video requires 30 frames per second. A 90 minute animated feature film contains a total of approximately 130,000 frames. Commercial animation studios paint images onto celluloid sheets (*cel*s), using two to six overlays per frame. This cel technique saves some time because only the parts of an image which actually move in a particular frame have to be redrawn. Even so, the entire process can be very costly. A feature film may require the preparation of over half a million cels.

The first efforts to utilize digital computers in the production of animated films date back to the early 1960's. Recent advances in computer graphics hardware and software have made computer animation a rapidly expanding field which now includes a large number of different production styles, approaches, and techniques.

This chapter introduces some of the areas in traditional animation for which computer systems have been developed. The remainder of this thesis is mostly concerned with one particular aspect of cel animation, the generation of "inbetween drawings".

### 1.1. Cel Animation

The production of traditional cel animation is based on the pipeline principle and requires a large staff of trained artists and technicians. The overview below outlines some of the most important stages in the assembly line; a more complete description can be found in some of the animation books listed in the references.

One of the first steps in the production of animated films is the development of the *storyboard*, a graphic equivalent of the script in live-action cinematography. Before work on the animation can begin, it is usually necessary to record the soundtrack and analyze it carefully. Detailed bar sheets listing all words and important notes of



music are prepared to provide accurate timing information for the synchronization of sound and picture. Using the storyboard and the bar sheets the head animators create a set of key drawings (*extremes*) for each sequence in the film. These pencil sketches showing the characters at crucial points in the action are passed to the *inbetweener* who interpolates all the missing frames required to produce smooth animated motion.

The pencil drawings are then photographed to produce a black and white *line test* which is used to check the quality and dynamics of the animation. Once the line tests are approved an *inker* transfers the pencil sketches to sheets of celluloid. In many commercial studios this step has been automated, and the drawings are copied to acetate by xerography. Cels carrying the ink outlines are then passed to an *opaquer* who fills in the outlines with opaque paints according to a colour scheme supplied by the character designer or the head animator. A commercial studio may have several hundred inkers and opaquer working on different sequences in parallel to speed up the process.

Meanwhile, another group of artists is producing the background paintings for the film. Because the backgrounds are not animated (except for camera movements such as zooms or pans) these paintings can be quite detailed and visually far more complex than the animation itself. The last step is to combine the animation cels and background paintings on an *animation table* and to film them, one frame at a time, producing the final result.

Considering the amount of repetitive labour required, cel animation is a tedious process. The introduction of computers to reduce manual labour at various stages of the production pipeline seemed like a very natural idea in the early 1970's. It is still a subject of debate as to which parts of the pipeline can benefit most from computer assistance. The answer depends on the desired animation style, the production context, available resources, and many other factors. There are several different approaches to the problems of computer assisted inbetweening, inking, opaquer, painting, and filming which are worth examining.

### 1.1.1. Inbetweening

The area which was the focus of much of the early work in computer animation is the generation of inbetweens. Many different techniques have been developed for creating inbetweens based on either key drawings or other information. These systems can be divided into two major categories, those which are *language driven*, and those which are *picture driven*. In picture driven systems the animation is controlled through drawings, for example through keyframes between which the system interpolates. The animator might also draw curves which determine the path of movement for an object. In language driven systems the generation of inbetweens is specified by some form of written script or command language. For example, the animator might first construct a complex object by combining simpler predefined shapes. He could then animate it by specifying the amount of rotation, translation or scaling which is to be applied to the object over a given number of frames.

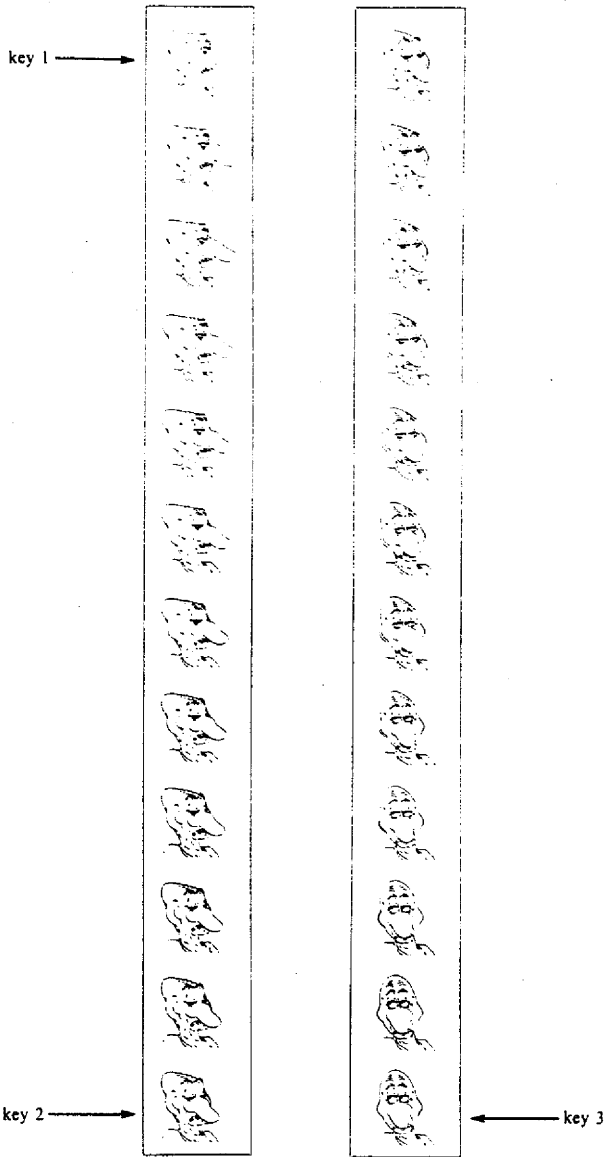


Figure 1.1 A sequence of inbetweens based on three keys

These different approaches work best for specific types of animation. Language driven systems are generally more suited for precise technical animation because shapes and movements can be defined mathematically. However, individual objects in these systems are usually rigid or can only be distorted mathematically. Consequently language driven systems are less suitable for character animation where subtle changes in facial expressions and metamorphosis of shapes is required. Picture driven systems, especially keyframe animation, are much more appropriate for this style of animation because the animator has direct intuitive control over the animation through his drawings. Figure 1.1 shows a one-second sequence produced from three key drawings with the keyframe animation system described in Chapter 2. In Figure 1.2 the animator uses an electronic *digitizing tablet* and *stylus* to enter a keyframe for a different part of the same sequence. Computer assisted inbetweening will be discussed in much more detail in Chapter 2.

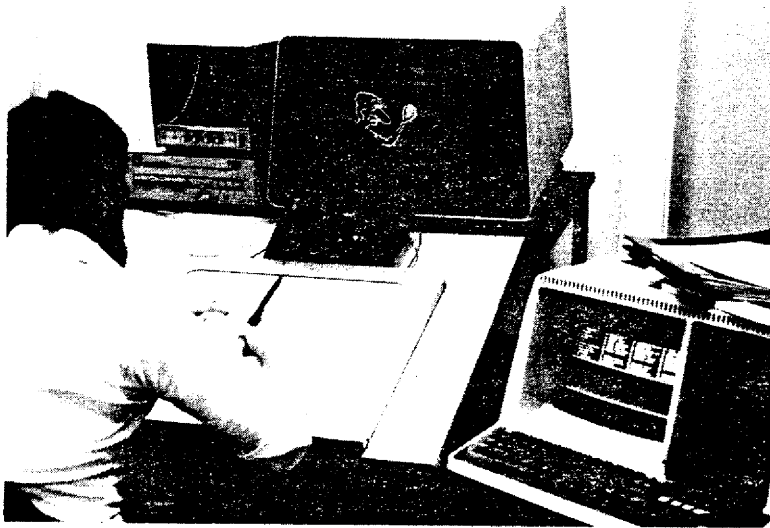


Figure 1.2 Animator entering drawings at work station

### 1.1.2. Inking, Opaquing, and Painting

In large commercial animation studios most of the production budget for a film is consumed by the inking and opaquing phases. For this reason some system designers have decided to leave the generation of inbetween drawings as a manual process while concentrating on introducing automation for inking and opaquing. A system of this type was developed at Hanna-Barbera studios in cooperation with Cornell University. Standard manual techniques are used up to the creation of outline drawings on paper for each frame in the sequence. Then, instead of transferring the drawings to cels by manual inking or xerography, they are scanned into the computer with a video camera. Image processing techniques are used to remove any "noise" from the scanned image, and the individual frames are now ready for electronic opaquing.

The opaquer sees the scanned image on a colour video display. With an electronic tablet and stylus he can select a colour from a palette displayed on the monitor and use it to fill any area on the screen by touching (*seeding*) some point inside the region and letting an *area flood algorithm* fill the region up to its enclosing boundary. Compared with manual techniques, these algorithms are very fast, reducing the opaquing time per cel by a factor of five or more. A similar system, "Scan'n'Paint", developed by Garland Stern at the New York Institute of Technology, has been used effectively in such productions as "Measure for Measure", a 22 minute animated explanation of the metric system.

In the meantime, other artists are preparing the backgrounds with computerized *painting systems* which allow them to use a colour monitor and a tablet with a stylus like a canvas and brush. In the simplest case, the system stores a trail of "paint" following the movements of the artist's hand on the tablet and displays the resulting picture on a colour monitor. Most painting systems are much more sophisticated, allowing for colour blending, simulation of air brushing, and other effects. One of the earlier systems of this type was PAINT, developed by Alvy Ray Smith at the New York Institute of Technology [Smith 78].

### 1.1.3. Merging and Filming

When all cels have been scanned in and opaqued and the backgrounds have been painted the final step is the assembly (*merging*) of all separate image components, the application of camera effects such as zooms and pans, and the recording onto video tape or film. Many of the constraints in traditional animation are a consequence of the equipment and techniques used in the assembly and filming stages. A conventional animation stand consists of a single-frame camera mounted vertically above a table which can be moved north-south and east-west as well as rotated. Artwork is attached to a set of registration pegs mounted on bars which can slide in an east-west direction on the table. With this type of equipment it is difficult to manipulate more than a few cel layers independently. In 1936 this problem led to the invention by Walt Disney of the *multiplane camera*. Here the artwork is placed on individually lit plates of glass mounted at different heights below the camera. Each level can be manipulated independently, so the technique creates a convincing illu-

sion of three dimensional space. Unfortunately, such a stand is expensive to construct and may require a dozen technicians to operate.

Computerized assembly and filming systems which can produce animation even beyond the capabilities of a multiplane camera have been developed by [Levoy 77] and [Wallace 81]. Because all zooms, pans and other transformations are applied mathematically before the final image is produced, these systems use standard film and video recorders with a fixed camera and picture position.

## 1.2. 3-D Animation

Traditional 3-D animation is created by constructing models of objects and characters which are moved within a scaled set and filmed, one frame at a time. The models can be extremely complex, often consisting of an articulate internal skeleton with a realistic external shell of plastic or latex rubber. Animating the models is a time consuming procedure because their positions usually change by only a fraction of an inch in every frame.

Computer animation techniques have helped to produce 3-D sequences for commercials and special effects which could not have been produced with any traditional models. In most 3-D computer animation systems the model construction and animation phases are treated separately. Models are usually constructed from primitive mathematical objects such as polygons or patches, or from basic solids such as spheres and/or cylinders. Texture and other detail may be added with a variety of techniques, and lighting models with shading and reflections provide a high degree of realism.

To manipulate the model, the animator usually works with a simplified skeletal version which he can position interactively. In this way he can construct a number of key poses between which the system can then interpolate. Once the animation is found to be satisfactory, the detailed model can be made to go through the same sequence of transformations as the skeleton to produce the final animation.

This simplified description of full 3-D animation is included only for completeness. A more detailed introduction can be found in [Booth et al. 82] or [Kovacs et al. 82]. The remainder of this thesis discusses a 2-D animation system with some limited 3-D capabilities which is currently in use at the French Animation Studio of the National Film Board of Canada.

## 2. THE "CINANIM" KEYFRAME ANIMATION SYSTEM

The National Film Board of Canada has been actively involved in computer animation since the early 1970's. The prototype system described in this thesis is a product of the experience the National Film Board has gained over the last decade. Before describing the system in detail, it is useful to take a brief look at the context in which this development took place.

### 2.1. Historical Background

In the late 1960's Nestor Burtnyk and Marcelli Wein at the National Research Council of Canada developed a computer animation system based on the keyframe technique. Over the next few years several NFB animators experimented with the system at the National Research Council in Ottawa.

Encouraged by these early results, the Research Council and the Film Board agreed to embark on a major film production. The resulting film "Hunger" ("La Faim"), directed by Peter Foldes [Foldes 73] and produced by Rene Jodoin, was a milestone in computer animation. It won much international acclaim, including the Special Jury Prize for Best Short Film at the Cannes Film Festival in 1973. The success of "Hunger" indicated that computer graphics could be a very powerful tool for animators at the Film Board.

In 1975 the National Research Council awarded a software development contract to Digital Methods Ltd. to extend the animation system and package it in a form suitable for use by computer-naive animators at the National Film Board's animation studio. In the meantime the Board purchased the necessary system hardware and developed a low cost colour film recorder for final production hardcopy. By 1979 software development had not been completed, but project funds had run out. The system was delivered to the Film Board in an "as is" state. Most of the software had been written [DML 79], but it soon became obvious that the debugging and testing phases had only just begun when system installation was completed in September 1979.

After several months of experimentation and frustration it became clear that further software development and maintenance were needed to turn the system into a viable production tool. Over the next two years all of the nine major user processes in the system were modified to some extent. In some cases only minor changes in the user interface have been made, in other cases new functions have been added. The VIDEO and FILM processes have been rewritten almost completely.

Section 2.2 describes the animator's view of this revised system. Even though a number of problems still remain, this system is currently in use producing several

animated short films. Section 2.3 describes the programmer's view of the INTERPOLATION process, which is the core of any keyframe animation system. The INTERPOLATION process has been completely rewritten around a new algorithm which generates smooth motion. This Curved INterpolation ANIMATION system ("CINANIM") is currently operational in a prototype version. A short test film (Appendix D) demonstrates the feasibility of the new algorithm. Certain features such as hidden line removal and area fill have not been implemented in the new interpolation process, and therefore the current production system at the Film Board still uses the old linear interpolation scheme.

## **2.2. The User's View**

This section describes the animation system from the user's point of view. The description is not intended to be a users' manual, thus many of the system's functions will not be described in detail. The goal is to provide some insights into the underlying logical structure and to outline the roles of the different system processes in the production of an animated film.

### **2.2.1. Operating Environment**

From the animator's point of view, the system hardware can be grouped into three major functional units, as shown in Figure 2.1.

The processing subsystem consists of a PDP-11/34 minicomputer, a disk drive, a tape drive, and a system console. The animator only interacts with this part of the system to load his disk and boot the system at the beginning of a session and to mount tapes during a session.

For most of his work, the animator uses the various input and output devices which constitute the animator's workstation. The principal input device is a digitizing tablet with an electronic stylus which is used to enter drawings into the system and to control system operation through menu selection (see Section 2.2.2). The principal output device is a calligraphic display which is used to display drawings as they are entered into the system and to play back simple animation sequences. An alphanumeric terminal displays the current menu and allows input of textual information such as picture names. The functions of the other devices which are part of the animator's workstation shown in Figure 2.1 will be described in more detail in Section 2.2.2.

The third hardware subsystem is the filming station which consists of a high precision cathode ray tube, a filter wheel, and an animation camera. The animator's interaction with this equipment is limited to loading and unloading the camera and to running a preliminary check ensuring that the hardware is operating correctly. The filming station is located in the same room as the processing subsystem while the animator's workstation (Figure 1.2) occupies a small adjacent room.

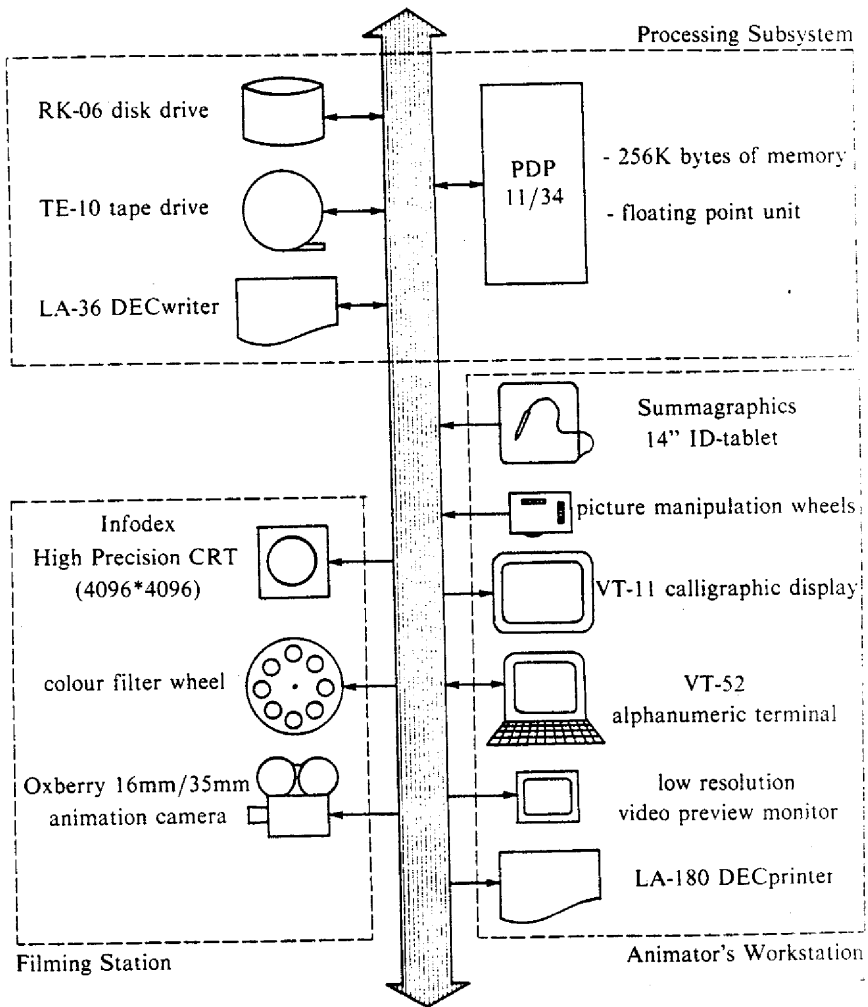


Figure 2.1 Animation system hardware configuration



### 2.2.2. System Description

From the user's point of view, the system is divided into nine major processes:

- DRAW
- MANIPULATE
- GENERATE
- SKELETON
- COMPOSE
- INTERPOLATE
- VIDEO
- FILM
- UTILITY

The above division does not correspond exactly to the underlying software structure because in some cases several of these user processes have been combined into a single software task while in other cases a single user process has been implemented as several separate system tasks.

To initialize the system at the beginning of a session the animator loads his disk, boots the processor, and types on the alphanumeric terminal the name of a command file. This command file installs all of the necessary system tasks, opens all required files, selects the specified interaction language (the system is available in English or French), and activates the DRAW process.

From then on, system operation is entirely controlled through the use of *menus*. A menu is a list of options from which the user selects the next action he wants the system to execute. The animation system has nine different menus, one for each of the user processes. The current menu (initially the DRAW menu) is displayed in the upper half of the alphanumeric terminal screen (see Figure 2.2). Each menu item is

menu area	DRAW	10 COPY REF	20 COMBINE CELS	30 LIST PIC ON/off
	1 DRAW	11 COPY STROKE	21 MOVE REF	31 SAVE PICTURE
	2 coarse	12 BLANK STROKE	22 RESTORE REF	32 REPLACE PICTURE
	3 MEDIUM	13 CONT STROKE	23 STEP REF	33 SAVE CEL
	4 fine	14 LOCK STR OFF/on	24 RE-OPEN A CEL	34 REPLACE CEL
	5 rubber band	15 CLOSE OUTLINE	25 NEW CEL	35 GET PICTURE/CEL
	6 rubber grid lock	16	26	36 DESTROY PICT/CEL
	7 SCROLL	17 OUTLINE/detail	27 ERASE STROKE	37
	8 compare	18 COMTIG OFF/on	28 ERASE REF	38 GET REF
	9 3D OFF/on	19 REGISTER	29 ERASE CEL	39 RESTART
table area	.....			
	NUMBER OF ITEMS IN PICTURE FILE #1 = 5 (PAGE 1)			
	1 HEAD1	10	19	28
	2 ARMS1	11	20	29
	3 BODY1	12	21	30
	4 LEGS1	13	22	31
	5 HEAD2	14	23	32
	6	15	24	33
	7	16	25	34
	8	17	26	35
prompt line	9	18	27	36
	.....			
message line	ENTER NAME OF PICTURE TO BE DELETED: ARMS2			
	[107] THE SPECIFIED PICTURE DOES NOT EXIST!			

Figure 2.2 Display format for alphanumeric terminal

labeled with a number from 1 to 39. To select an item the animator depresses the stylus on the corresponding tablet "button" which is marked on an acetate overlay attached to the tablet (Figure 2.3). The STOP button in the upper left corner of the tablet can be activated at any time and will terminate the current session if the animator reconfirms his intention to do so. In addition to the 39 menu buttons which occupy the upper portion of the tablet, another set of buttons is marked on the overlay. Nine of these buttons correspond to the user processes mentioned earlier. When one of these areas is selected, the system switches to the requested process and displays the associated menu on the alphanumeric terminal.

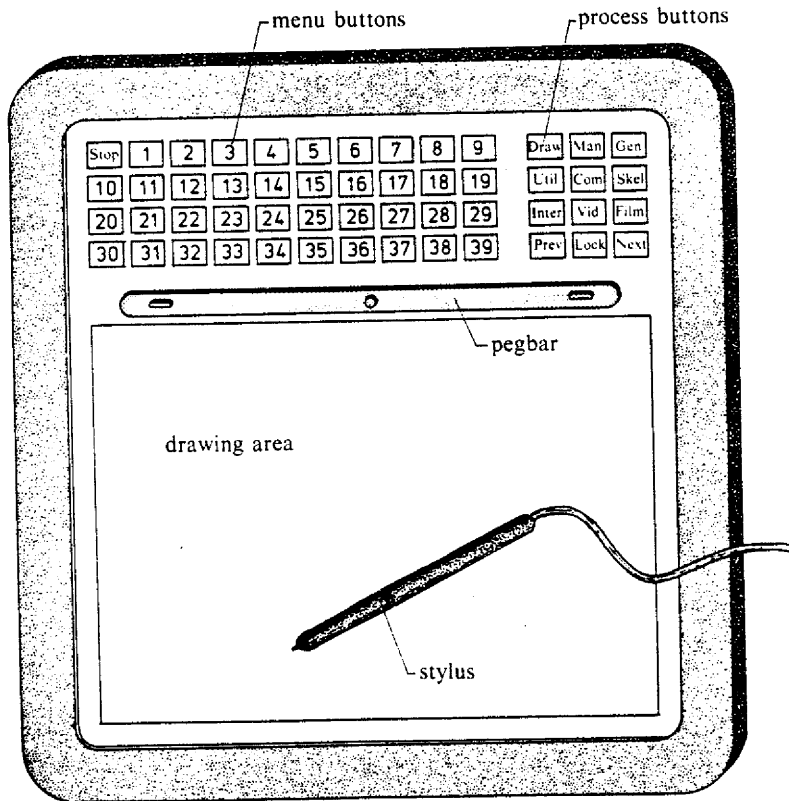


Figure 2.3 Digitizing tablet with menu overlay

Figure 2.4 gives an overview of the normal sequence of steps involved in the production of an animated film with this system. First the animator enters the DRAW process to trace all of the keyframes and store them in a *picture file*. If

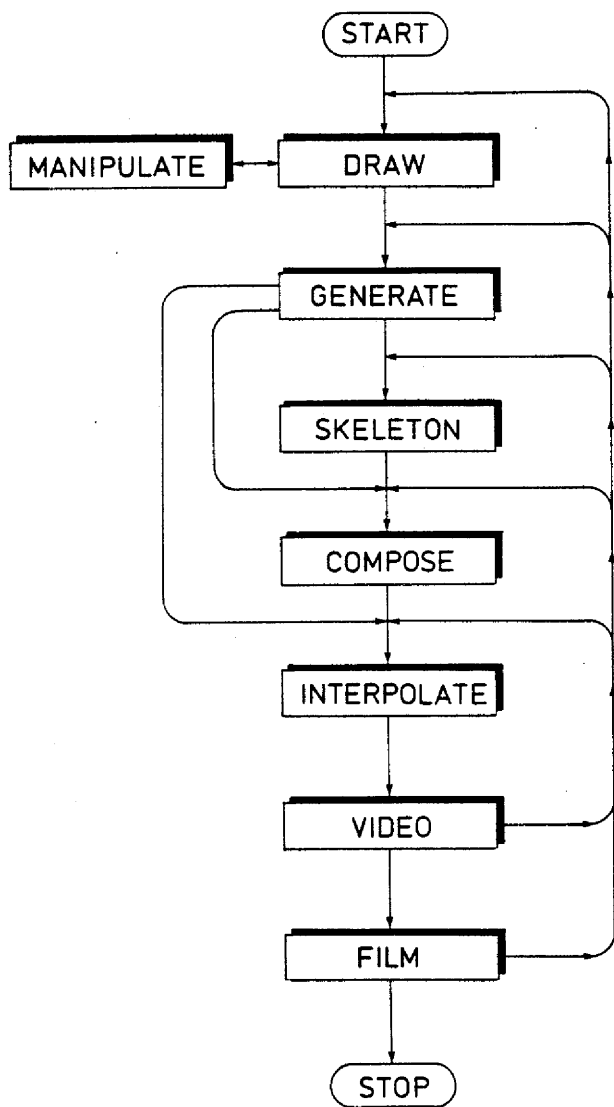


Figure 2.4 Overview of steps required to produce an animated film

necessary he may occasionally switch to the MANIPULATE process to modify pictures by scaling, translating, or rotating them. The animator then selects the GENERATE process which allows him to generate a *sequence file* by specifying the keyframes, the number of inbetweens for each interval, and other control information. If necessary he may also use the SKELETON process to construct a stick figure sequence to control the motion. For complex animation, several sequences and skeleton control sequences may be combined into a single *composition file* in the COMPOSE process.

Once the sequence or composition has been constructed, the animator enters the INTERPOLATE process which generates all the inbetweens based on the specified keyframes. Usually the result of this interpolation is first previewed in the VIDEO process. Once any required corrections have been made, the final colour output is produced in the FILM process. After viewing the film, it may be necessary to repeat one or more of the earlier steps to correct any flaws which were not visible in the low resolution black and white video preview. The UTILITY process may be invoked at any time to provide basic housekeeping functions such as transferring files between disk and tape or printing files and directories. The remainder of this section describes each of the nine user processes in more detail.

### 2.2.2.1. DRAW

The DRAW process allows the animator to enter the keyframes into the system using the digitizing tablet. A large percentage of the animator's interaction with the system is spent in this process. Before describing some of the DRAW functions, a brief explanation of the conceptual model underlying the structure of pictures within the system is necessary.

A *picture* is a compound block of visual information which consists of one or more *cels*. The term "cel" is derived from standard animation terminology where sheets of acetate ("celluloids") are used to carry portions of a picture. To animate a character, its head may be drawn on one cel while its arms, body, and legs would be drawn on two other cels (Figure 2.5). This method greatly reduces the amount of repetitive drawing required for animation. If, for example, the animator wants to move the character's head, he can simply prepare a series of head cels without redrawing any of the other body parts. A cel in the animation system is further subdivided into a sequence of *strokes*. When keyframes are traced in, a stroke is started by depressing the stylus tip on the tablet and terminated by lifting it. The path of movement of the stylus during this period is stored as a set of closely spaced *sample points*. Thus a stroke actually consists of a sequence of short vectors.

The concept of a stroke is extremely important because the system interpolates between keyframes by using the strokes to establish a correspondence between the *source* and the *destination* cels. The first stroke from the source cel is gradually transformed into the first stroke of the destination cel, the second stroke to the second stroke, etc. Corresponding strokes do not need to have the same number of points because the system automatically adds intermediate points where necessary. If two keyframes do not have the same number of strokes, the system will break or merge strokes to match the two keys, but the results will usually be somewhat unpredictable.

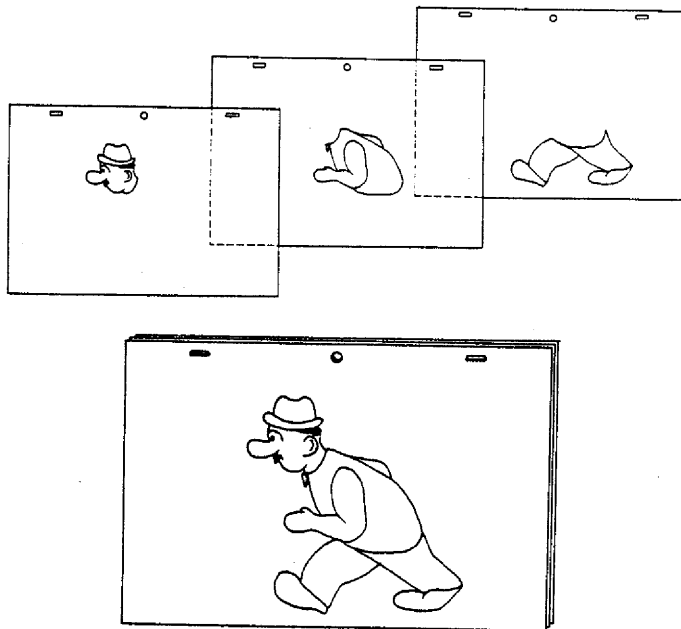


Figure 2.5 A picture consisting of three cels

In most cases the animator decides ahead of time how a drawing will be broken up into strokes and then uses the same ordering consistently for the entire sequence of cels. Figure 2.6 illustrates the importance of correct stroke ordering showing (a) the result of interpolating between two keys drawn with matching strokes, and (b) the result when the order of strokes in the destination cel has been changed. In the latter case, during interpolation the image degenerates between keys before reassembling itself as the destination key is reached.

Strokes can only be manipulated within the DRAW process because all other processes work at a conceptually higher level, dealing only with cels, pictures, or sequences. The DRAW menu (Figure 2.7) provides a large number of functions for working with strokes, cels, and pictures. It should be pointed out that a core of about a dozen selections is sufficient for entering most drawings. The remainder of this section describes the basic functions as well as some of the more specialized selections.

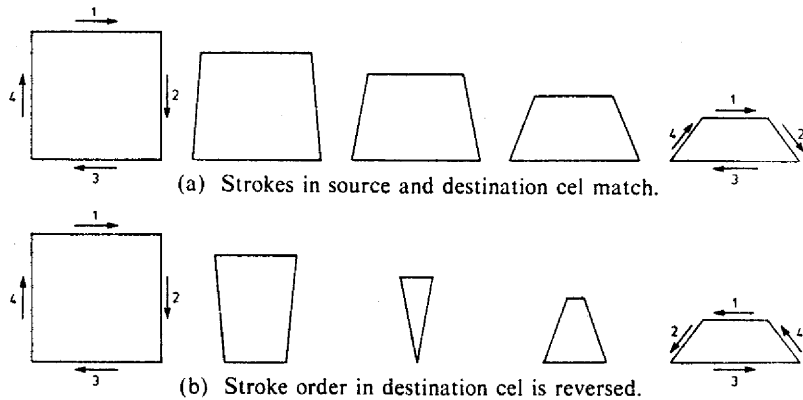


Figure 2.6 The importance of correct stroke ordering

The first six menu items are used to select the current drawing mode. For free-hand curve drawing COARSE, MEDIUM or FINE resolution can be requested. The active selection (highlighted in upper case letters within the menu) determines the number of sample points taken to represent a hand-drawn curve. MEDIUM is adequate for most purposes. For geometric pictures the RUBBER BAND function allows the user to draw straight lines by defining only endpoints. On the graphics display a line starting at the previous endpoint follows the cursor movement for continuous feedback (like a rubber band attached to the stylus) until the stylus tip is depressed to enter the next endpoint. RUBBER GRID LOCK works in a similar manner, except that line endpoints are constrained to lie on a predetermined grid.

DRAW	10 COPY REF	20 COMBINE CELS	30 LIST PIC ON/off
1 DRAW	11 COPY STROKE	21 MOVE REF	31 SAVE PICTURE
2 coarse	12 BLANK STROKE	22 RESTORE REF	32 REPLACE PICTURE
3 MEDIUM	13 CONT STROKE	23 STEP REF	33 SAVE CEL
4 fine	14 LOCK STR OFF/on	24 RE-OPEN A CEL	34 REPLACE CEL
5 rubber band	15 CLOSE OUTLINE	25 NEW CEL	35 GET PICTURE/CEL
6 rubber grid lock	16	26	36*DESTROY PICT/CEL
7 SCROLL	17 OUTLINE/detail	27 ERASE STROKE	37
8 compare	18 CONTIG OFF/on	28 ERASE REF	38 GET REF
9 3D OFF/on	19 REGISTER	29 ERASE CEL	39 RESTART
*****			
NUMBER OF ITEMS IN PICTURE FILE #1 = 5 (PAGE 1)			
1 HEAD1	10	19	28
2 ARMS1	11	20	29
3 BODY1	12	21	30
4 LEGS1	13	22	31
5 HEAD2	14	23	32
6	15	24	33
7	16	25	34
8	17	26	35
9	18	27	36
*****			
ENTER NAME OF PICTURE TO BE DELETED: ARMS2			
[107] THE SPECIFIED PICTURE DOES NOT EXIST!			

Figure 2.7 The DRAW menu

The 3D ON/off switch (menu item #9) determines the number of coordinates to be stored for each sample point in the current cel. In 3D mode, the system stores a z-coordinate of 0 for each point since the tablet only provides (x,y) coordinates. The z-coordinate can later be changed through 3D translations, rotations, and scaling in the MANIPULATE process. Turning the 3D mode off for pictures where it is not required reduces storage space requirements by about 30%.

Even though cels are displayed only as vector images on the VT-11 graphics monitor, the animator can specify areas to be filled with solid colours when the sequence is filmed. For this purpose the system distinguishes between OUTLINE and DETAIL strokes (menu item #17). A DETAIL stroke is a sequence of vectors which does not delineate any surface, while a set of OUTLINE strokes implicitly forms the contour of a filled area. When all OUTLINE strokes enclosing an area have been drawn, the animator selects the CLOSE OUTLINE function (menu item #15). The system then asks him to type in a *colour code* (a number in the range from 0 to 63) for this area.

The last stroke in the *working cel* (the cel currently being drawn) can always be deleted with the ERASE STROKE function (menu item #27). Selecting this function several times will delete the last few strokes in the cel, but it is currently not possible to erase an arbitrary stroke in the cel. The entire working cel can be deleted with the ERASE CEL function (menu item #29). The last column in the menu provides functions which let the animator save or replace individual cels or pictures in a *picture file* or call up cels or pictures which have previously been stored. The GET REF function (menu item #38) allows the animator to call up a *reference cel* from the picture file which will be displayed at low intensity. As the animator enters strokes for the current cel, the corresponding strokes in the reference cel are highlighted in turn to show which strokes will be matched up when the system interpolates between these two cels. Reference cels are also useful because individual strokes or the entire cel can be copied into the current working cel with the COPY STROKE and COPY REF functions (menu items #11 and #10). Finally, the RESTART function (menu item #39) erases all cels, including references, from the display.

### 2.2.2.2. MANIPULATE

The MANIPULATE menu (Figure 2.8) allows the animator to modify existing cels and pictures through the use of geometric transformations. To reduce the frequency with which the animator has to switch between menus, selections #28 through #39 from the DRAW process have been duplicated in the MANIPULATE menu. Depending on the TAGGED/all selection (menu item #1), transformations may be applied to all displayed cels or only to those which have previously been selected with the stylus in SET TAGS mode (menu item #2). All transformations can be specified through either digital or analogue input, depending on the setting of the DIGITAL/wheel switch (menu item #11). The *picture manipulation wheels* shown in Figure 2.9 are used for analogue input. In general each of the three wheels controls the current transformation for one coordinate axis. For example, when MOVE (menu

1  
2  
3  
4  
5  
6  
7  
8  
9  
\*\*

1  
2  
3  
4  
5  
6  
7  
8  
9  
\*\*  
EC

it  
tl  
d  
c  
F  
F  
#  
c  
v

MANIPULATE	10	20 COMBINE CELS	30 LIST PIC ON/off
1 TAGGED/all	11 DIGITAL/wheel	21 PERSPECTIVE	31 SAVE PICTURE
2 SET TAGS	12	22 CLEAR PERSPECT.	32 REPLACE PICTURE
3 GRID ON/off	13	23	33 SAVE CEL
4 BORDER OFF/on	14 CLEAR ALL	24 TRANSFORM ALL	34 REPLACE CEL
5 move	15 CLEAR MOVE	25 TRANSFORM MOVE	35 GET PICTURE/CEL
6*ROTATE	16 CLEAR ROTATE	26 TRANSFORM ROTATE	36 DESTROY PICT/CEL
7 size	17 CLEAR SIZE	27 TRANSFORM SIZE	37
8 stretch	18	28 ERASE REF	38 GET REF
9 mirror	19 COPY MATRIX	29 ERASE CELS	39 RESTART
*****			
NUMBER OF ITEMS IN PICTURE FILE #1 = 5 (PAGE 1)			
1 HEAD1	10	19	28
2 ARMS1	11	20	29
3 BODY1	12	21	30
4 LEGS1	13	22	31
5 HEAD2	14	23	32
6	15	24	33
7	16	25	34
8	17	26	35
9	18	27	36
*****			
EQUIV. ROTATION: X=- 35.0, Y= 27.6, Z= 120.8; NEW ROTATION (DEG./AXIS): _			

Figure 2.8 The MANIPULATE menu

item #5) has been selected as the current transformation, turning Wheel 1 will move the picture horizontally, Wheel 2 moves it vertically, and Wheel 3 changes the z-depth. At the same time, a set of counters on the alphanumeric terminal indicates the current translation values in 1/100". If a cel has been moved by mistake, it can simply be restored by the CLEAR MOVE selection (menu item #15). Once the desired position for a cel has been found, the TRANSFORM MOVE function (menu item #25) makes the translation permanent by multiplying each point in the cel by the current translation matrix and resetting the matrix. Now a CLEAR MOVE selection will not have any effect.

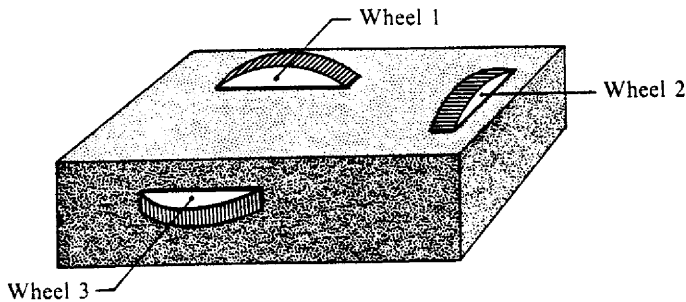


Figure 2.9 Picture manipulation wheels



The ROTATE, SIZE, STRETCH, and MIRROR functions (menu items #6-#9), together with the corresponding CLEAR and TRANSFORM operations, are used in a similar manner. STRETCH is a generalized scaling function which allows independent scale factors for the x, y, and z-axis. SIZE provides uniform scaling for all three axes, whereas MIRROR scales all x-coordinates by -1. In DIGITAL mode, rotation angles, scale factors, and translation offsets are specified numerically in response to a system prompt on the alphanumeric terminal. Digital and wheel input mode are completely compatible, and the animator can switch freely between them. The alphanumeric terminal always displays the current transformation values, taking into account all manipulations (digital or analogue) since the last CLEAR or MOVE command. In practice the wheels are mostly used to manipulate a cel visually, but once the desired position is known, other cels in the same picture are usually manipulated digitally.

This approach presents a problem in the display of counters for analogue rotations. Since 3-D rotations are not commutative, simple cumulative counters have no real meaning unless the animator does all x-rotations first, then all y-rotations, and finally all z-rotations. Unfortunately, animators normally use all three wheels more or less simultaneously to obtain the desired rotation angles. To provide animators with a simple means of reproducing exactly the the same result digitally for other cels, the system displays a set of *equivalent rotation counters*. If the animator uses these three values in the order of x, y, z to specify a rotation digitally, the compound rotation will be exactly equivalent to the combined effect of all previous analogue rotations.

When several cels are to be transformed in the same way (e.g. all cels constituting a picture), the COPY MATRIX function (menu item #19) can be used to automatically apply a given sequence of transformations to a number of cels and save the transformed cels in the picture file.

### 2.2.2.3. GENERATE

Once all the keyframes for a sequence have been stored in the picture file, the animator enters the GENERATE process to construct the sequence. The GENERATE menu (Figure 2.10) provides various editing functions for *keyframe tables* (menu item #3) and *camera tables* (menu item #4). For both of these tables the animator can append new text lines to the end with the ENTER function (menu item #1), change existing lines with the UPDATE function (menu item #10), or INSERT (menu item #11) new lines anywhere in the table.

A typical example of a keyframe table is shown in Figure 2.10. For each keyframe in the sequence, the animator enters the names of all pictures or cels which must be included in that frame and the number of inbetween frames which the system must generate between this and the previous key. For each cel in the keyframe the animator can also optionally specify a *level* which is used to determine the priority of cels in the same frame for purposes of hidden surface removal. In addition, the animator can change the default *interpolation law* from linear (constant change in each frame) to obtain different dynamics such as accelerations, decelerations, or combinations of the two. Specification of a *skeleton* allows the motion of a

1	GENERATE	10 update	20 REPLACE CEL	30 LIST SEQ
1*	ENTER	11 insert	21 SUBSTITUTE DATA	31
2		12	22	32
3	KEYFRAME TABLE	13	23	33 SAVE SEQ
4	camera table	14	24	34 REPLACE SEQ
5		15 REPEAT FRAME	25 SWITCH PIC FILE	35 GET SEQ
6		16 COPY FRAME	26	36 DELETE SEQ
7		17	27	37
8		18	28	38
9		19	29	39 RESTART
*****				
LINE	KF NO	PICTURE	CEL	FRAMES
				CUM FR
1	3			12
2		BOY3		31
3			HEAD3	
4			BODY3	0
5			LEGS3	1
6		CAR3		2
7			CAR3	3
8	4			ACC1
9				10
				41
*****				
9				

Figure 2.10 The GENERATE menu

cel to be controlled by a simple stick figure (see Section 2.3.2.). Finally, a limited number of *special effects* can be requested. An example is the RUN-IN function which causes the image to appear gradually, as if it were tracing itself over the specified number of frames.

The camera table is used to describe simple camera effects. *Zoom factors* and *pan values* can be specified to enlarge or reduce the image and to move it horizontally or vertically. For each effect a start frame, a length in frames, and an interpolation law are specified. Again the default law is linear, but accelerations or decelerations can be requested.

When the sequence table and, optionally, the camera table have been entered, the animator stores the sequence on disk by selecting the SAVE SEQ function (menu item #33). Note that the last column of the GENERATE menu contains almost the same functions as the DRAW and MANIPULATE menus, except that here the operations apply to keyframe sequences.

#### 2.2.2.4. SKELETON

The SKELETON process allows the animator to control complex motion with simple stick figures, thereby reducing the number of detailed drawings that have to be prepared. The stick figures are entered in the DRAW process, using the RUBBER BAND mode, and stored in the picture file as *skeleton cels*, identified by a '\$'-prefix.

Using the SKELETON process, the animator constructs a *skeleton sequence* from these stick figure cels in much the same way as he would create a keyframe sequence in the GENERATE process. In addition to the functions for editing the

skeleton sequence table, the SKELETON menu (Figure 2.11) incorporates some transformation functions similar to those found in the MANIPULATE process (MOVE, ROTATE etc.) and a LINE TEST function for interpolating and previewing only the skeleton. The last column of the menu again contains the same functions as in the GENERATE menu, but here the operations apply to skeleton sequences.

SKELETON	10 UPDATE SET	20 SKIP	30 LIST SEQ					
1 ALL/selected	11 INSERT SET	21	31					
2 SELECT	12 KEYBOARD EDIT	22 BOUNDARY OFF/on	32					
3 HOOK	13 CONTROL FRAME	23 INBETWEEN ON/off	33 SAVE SEQ					
4 DRAG	14 CANCEL CTRL PRM	24	34 REPLACE SEQ					
5 MOVE	15 RENUMBER	25 SHIFT	35 GET SEQ					
6 ROTATE	16	26 SPREAD	36 DELETE SEQ					
7 SIZE	17 CHOOSE DRIVE	27 SHIFT/SPREAD OFF	37					
8 STRETCH	18 LINE TEST	28 ERASE REF PIC	38 GET REF PIC					
9	19	29	39 RESTART					
*****								
LINE	SET	FRAMES	CUM.F.	DRIVE	START-CEL	END-CEL	CNTRL-CEL	CNTRL-F
1	2	50	90					
2				1	ARMA	ARMD		
3							ARMX	1
4							\$	20
5							ARMC	49
6				2	LEGE	LEGC		
7								
8								
9								
*****								
7								

Figure 2.11 The SKELETON menu

The use of skeleton controls is probably the most advanced and powerful feature in the animation system, but unfortunately the current user interface for the SKELETON process is extremely cumbersome, and the skeleton concept has not been well integrated with the other parts of the system. For these reasons skeletons are only used infrequently in the current system. A detailed explanation of how skeleton sequences are constructed is beyond the scope of this section, but the skeleton concept itself is described further in Section 2.3.2.

### 2.2.2.5. COMPOSE

The COMPOSE menu (Figure 2.12) allows the animator to construct a *composition sequence* which may consist of up to 16 keyframe and/or skeleton sequences running in parallel or consecutively. For example, an animator might construct two keyframe sequences for the background scenery, two sequences to animate different characters, two skeleton sequences to control their motions, and another keyframe sequence for the foreground. In the COMPOSE process he could then type in a *sequence table* (Figure 2.12) which describes how the two skeleton and five keyframe sequences are to be combined into a single composition.

COMPOSE	10*UPDATE	20 LIST COLOURS	30 LIST COMP
1 enter	11 insert	21	31
2	12	22	32
3 SEQUENCE TABLE	13	23	33 SAVE COMP
4 camera table	14	24	34 REPLACE COMP
5 colours table	15	25	35 GET COMP
6	16	26	36 DELETE COMP
7	17	27	37
8	18	28	38
9	19	29	39 RESTART
*****			
LINE	START	END	SEQUENCE
1	0	399	FOREGR
2	0	399	MAN
3	0	299	\$MANSKEL
4	0	399	BOY
5	0	399	\$BOYSKEL
6	0	149	BACKGR1
7	150	399	BACKGR2
8			
9			
*****			
3	0	299	\$MANSKEL

Figure 2.12 The COMPOSE menu

As in the GENERATE process, a camera table can be entered to specify zooms and pans. The difference between the two camera tables is that a keyframe sequence camera table applies only to that particular sequence, while a composition camera table affects all sequences in the composition. Any sequence camera effects are applied first, thus allowing the animator to use both tables simultaneously to obtain separate effects for individual sequences as well as a global effect for the entire composition.

The editing procedures in the COMPOSE process are almost the same as those in the GENERATE process, and the two menus are similar.

### 2.2.2.6. INTERPOLATE

The INTERPOLATE process performs the central function of the animation system, the generation of inbetween frames based on the keyframes in a sequence. The animator first sets the KEYFRAME/compose switch (menu item #18, Figure 2.13) to specify whether he wants to interpolate a single keyframe sequence or a composition. He then selects the desired output type from menu items #2-#5. A LINES interpolation will produce vector data with all lines visible, while a HIDDEN LINES interpolation will output only those lines which are not hidden behind surfaces closer to the observer. An AREAS interpolation outputs the necessary raster line segments to fill all areas in the picture which are not hidden (Figure 2.14). The LINES + AREAS function has not been implemented, but will eventually produce data tapes containing both vector and area fill data, with hidden lines and surfaces removed. Section 2.2.2.8 explains how the line and area data is combined during the filming process.

INTERPOLATE	10	20 SKIP OFF/on	30
1	11	21	31
2 lines	12 line test	22	32
3 hidden lines	13	23	33
4 AREAS	14 TAPE	24 BK-GROUND ON/off	34
5 lines + areas	15	25	35
6	16 dec20 preprocess	26 PAUSE OFF/on	36
7	17	27	37
8*START	18 COMPOSE/keyframe	28	38
9	19	29	39 RESTART
*****			
NAME OF COMPOSITION: DYNOSAUR3		TOTAL FRAMES: 300	
START FRAME: 101			
LAST FRAME: 300			
CURRENTLY PROCESSING FRAME: 51			
TOTAL INTERPOLATION TIME SO FAR (HH:MM): 02:10			
ESTIMATED TIME REQUIRED TO FINISH (HH:MM): 06:30			
*****			

Figure 2.13 The INTERPOLATE menu

If the animator wants to interpolate a simple keyframe sequence in LINES mode, he can choose to preview it on the VT-11 display instead of writing the inbetween frames to tape by selecting the LINE TEST function (menu item #12) instead of the TAPE function (menu item #14). However, for most sequences the display speed in LINE TEST mode is far too slow, and the VIDEO process (Section 2.2.2.7) is used instead. The SKIP ON/off function (menu item #20) is often used to get a rough idea of slow animation and camera effects without spending a lot of time interpolating the entire sequence. If the SKIP option is turned on and a skip count of 4 has been specified, the system will skip four frames after every frame, calculating only frames 1,6,11,16 etc. The interpretation of the skip count is somewhat counterintuitive. To calculate frames 1,6,11,16 etc. it would be more logical to specify a skip count of 5 (interpolate every fifth frame).

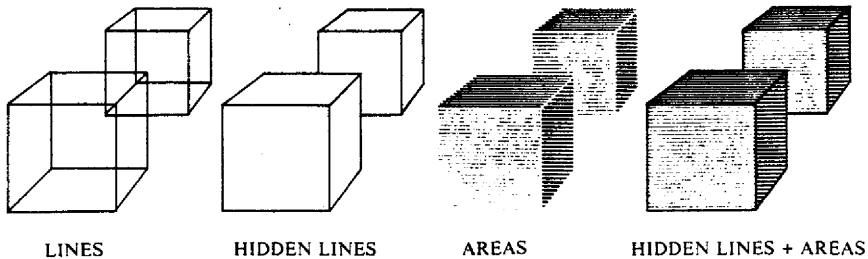


Figure 2.14 Available output types

After all desired options have been set up, the animator selects the START function (menu item #8) and responds to a series of system prompts requesting the sequence or composition name, the start frame, and the end frame. Once the animator confirms that he has mounted an output tape, interpolation begins and a number of counters on the VT-52 terminal (Figure 2.13) indicate the frame currently being processed, the amount of clock time that has elapsed so far, and an estimate of the time required to finish interpolating the sequence. Due to some extremely inefficient algorithms in the hidden surface processing code, interpolation of area data may take over five minutes per frame for complex sequences. It is hoped that the introduction of more efficient algorithms in the new interpolation process described in Section 2.3 will shorten this time considerably. Since no animator interaction is required once the interpolation has started, long interpolations are usually treated more like batch jobs and are started up at night by the last system user.

### 2.2.2.7. VIDEO

In the VIDEO process, the animator can preview the result of a LINES or HIDDEN LINES interpolation before committing it to film. The key concept here is to provide *realtime playback*. The realtime requirement is crucial for judging the correctness of the animation, but the LINE TEST function in the INTERPOLATE process usually displays images, except for very simple ones, at much less than the standard film frame rate of 24 frames per second. To obtain realtime playback for complex keyframe sequences and compositions, the interpolation calculation must be done ahead of time because it may take from a few seconds up to a few minutes per frame. In addition the display process must be made independent of the image complexity. This immediately eliminates the VT-11 calligraphic display because the display file for this device resides in main memory and is only large enough for a few thousand short vectors (above about 2000 short vectors, the display becomes noticeably dimmer and soon begins to flicker objectionably).

The solution to these problems is to convert the interpolated high resolution vector images into low resolution raster images which take a constant amount of time to display, independent of their complexity. Again, the calculation time for scan conversion is too high to allow conversion and display to happen "on the fly". Therefore the VIDEO process consists of two distinct phases, the PROCESS phase and the DISPLAY phase (menu item #2, Figure 2.15). To preview a sequence, the animator loads the tape containing the interpolated line data, selects the PROCESS phase, and then STARTS (menu item #8). The system will now read vector data from the tape one frame at a time and scan convert it to a bi-level raster format (resolution 256\*192\*1). These raster images, requiring 6K bytes of storage each, are written to a workfile on disk which can store up to 300 frames.

Once the entire sequence (or 300 frames of it) has been scan converted, the system automatically enters the DISPLAY phase. Now the raster images are sequentially read in from disk and transferred to the video preview monitor through a special interface.

VIDEO	10	20	30
1	11	21	31
2 DISPLAY/process	12	22	32
3	13	23	33
4	14	24	34
5 PROC SKIP OFF/on	15	25	35
6	16	26	36
7	17	27	37
8*START	18	28	38
9	19	29	39

\*\*\*\*\*

START FRAME: 1	USE FUNCTION KEYS TO CONTROL DISPLAY:
LAST FRAME: 599	>> NORMAL SPEED FORWARD
	> SINGLE FRAME FORWARD
CURRENTLY DISPLAYING FRAME: 235	< SINGLE FRAME REVERSE
	<< NORMAL SPEED REVERSE
	Q QUIT






\*\*\*\*\*

Figure 2.15 The VIDEO menu

In spite of the low spatial resolution of the images, a read/display cycle for a frame still requires more than 1/24 second and therefore the temporal resolution had to be lowered. A rate of 12 frames per second is convenient because the standard video rate of 60 fields per second can be divided exactly into 12 frames by displaying each image for five fields, using standard video sync. Because each frame is displayed for 1/12 second, this frame rate would slow down the animation by a factor of two. To compensate only every other frame from the tape is processed and displayed. In the first second of playback time 12 frames are shown, either frames 1,3,5...23 or frames 2,4,6...24. Thus the workfile which can store up to 300 raster images contains either the odd frames from 1 to 599 or the even frames from 2 to 600. In either case this allows a continuous playback of 25 seconds of realtime animation.

Given the low spatial resolution of the image, the 12 frames per second display rate is not very noticeable. For some sequences which require precise timing, for example sequences which have been synchronized to a pre-recorded soundtrack, the animator may need to preview the sequence twice, once for odd frames and once for even frames, to ensure that no single frame errors have slipped through. In this case the animator simply processes the sequence a second time, this time specifying a start frame of 2 in response to the initial system prompt.

The user interface for the VIDEO process has been completely redesigned to allow the animator to interact naturally and immediately with the animation playback. Once the system has entered the display phase, the first frame in the sequence is displayed on the video monitor. The system then pauses to accept input from the keyboard. From this point on the display phase is controlled by the use of five "function buttons" (in fact these are ordinary keys on the VT-52 auxiliary numeric keypad which have been relabeled). The meaning of the five keys is shown on the VT-52 display (Figure 2.15). The keys are labeled:

	display sequence at normal speed, forward direction
	display sequence in single frame mode, forward direction
	display sequence in single frame mode, reverse direction
	display sequence at normal speed, reverse direction
	quit (exit from display phase)

These functions buttons are the only part of the animation system where keyboard input is in "raw" mode, i.e. not terminated by a carriage return, thus allowing the animator to place four fingers over the function buttons (which are adjacent) and use them like speed and direction controls on a cassette deck. In normal speed mode the playback continues until the end of the sequence is reached unless another function key is pressed in the meantime. In single frame mode a new frame is displayed each time the key is pressed. Several counters on the VT-52 terminal show the number of the frame currently being displayed as well as the start frame and end frame number for the sequence in the workfile. By running back and forth through the sequence and by looking at individual frames, the animator can catch most of the errors in sequences interpolated in line mode before actually filming them.

### 2.2.2.8. FILM

The FILM menu (Figure 2.16) provides the necessary control functions for the animation camera, filter wheel, and high precision CRT to produce the final hardcopy of an interpolated sequence. The animator first selects the appropriate output type (menu items #2-#5) depending on the type of sequence he wants to film. The four output types are described in more detail below. In the simplest case he then selects START (menu item #8), at which point the system will ask him to mount the data tape and specify the first and the last frame to be filmed.

More advanced special effects options allow the animator to run the camera in the REVERSE direction (menu item #23), to SKIP frames (menu item #20) on film by not exposing them, or to expose the film in MULTI FRAME mode (menu item #21), repeating each frame from the tape several times on film. In addition some of the functions which are available on the camera control panel can also be activated through menu selections. These operations include REWIND FILM (menu item #34) and POSITION FILM (menu item #32). The PHOTOCCELL selection (menu item #39) activates a test program which repeatedly fills a small rectangle in one corner of the high precision CRT display screen. The light intensity output is measured by a photocell glued to the face of the display and a voltmeter attached to the filming station. Before starting to film, the animator usually runs this test and, if necessary, adjusts the brightness control of the high precision CRT to bring the voltmeter reading to an established level.



FILM	10	20 EXP SKIP ON/off	30 LEADER OFF/on
1	11	21 SINGLE/multi FRM	31
2 colour line	12	22	32 POSITION FILM
3 black line	13 FILM	23 FILM FORWARD/rev	33
4 COLOUR AREA	14 display & film	24	34 REWIND FILM
5 black area	15 display	25	35
6	16	26 PAUSE OFF/on	36
7	17	27	37
8*START	18	28	38
9	19	29	39 PHOTOCCELL
*****			
START FRAME:	1	SKIP FACTOR:	1
LAST FRAME:	300		
CURRENTLY FILMING FRAME:	82		
FILMING STATION COUNTER A:	263		
FILMING STATION COUNTER B:	163		
*****			

Figure 2.16 The FILM menu

Once filming has started, the operator does not interact with the system any further, except possibly to abort the filming process. The filming program controls the film advance as well as the filter positioning mechanism. A set of counters on the VT-52 screen displays the current frame number in the sequence, the current frame number on film (which may be different), and other status information. In BLACK LINE mode (menu item #3) no filter is used, and all lines are filmed at a constant intensity on high contrast film, producing a negative with black lines on a colour background. In COLOUR AREA mode (menu item #4) three separate exposures are required for each frame. The system first selects the red filter and exposes all areas in the image which contain any red. The intensity for each area is determined by using the *colour lookup table* entry corresponding to the colour code for the area. Similarly, the green and blue components are exposed at the required intensities through the green and blue filters, before the film is finally advanced to the next frame.

Generally, the most pleasing final result is obtained by combining black outlines with coloured areas. This combination has been used for decades in traditional cel animation where black outlines are drawn with a fine ink pen (or, more recently, by xerography) on the front of the cel which is then flipped over and painted in various colours from the back. The black outlines make the image appear much crisper by separating the coloured areas slightly. They also save considerable time for the painter because small imperfections in the edge of a painted area will be hidden by the black lines on the front of the cel.

In computer animation, black outlines have similar advantages. They help to eliminate *bleeding* between adjacent colours. For example, if a red and a green area touch, minor inaccuracies in the CRT deflection and focusing circuits as well as

blooming on the CRT or the film itself often produce a distracting yellow line between the two areas. A second consideration is that vector calculations are usually faster than raster calculations, and in this system lines are generated at the full CRT resolution of  $4096 \times 3072$  while areas are filled at a much lower resolution of  $1024 \times 768$ . Jagged edges (*aliasing*) in the colour areas can effectively be hidden by covering them with high precision black outlines.

Unfortunately, it is not possible to expose black lines directly on top of the colour areas because the film registers light in an additive process, and therefore any area which has been exposed to light can only be exposed further but can never be restored to black. For this reason the system uses a two-pass approach called *bi-packing*. This technique is commonly used in the production of special effects and film titles as part of the optical printing process. First the outlines are exposed on high contrast film using the BLACK LINE mode. When the negative is developed it shows black lines on clear film. For the second pass a special bi-pack magazine (Figure 2.17) is mounted on the animation camera. Feed chamber #1 is loaded with raw colour negative stock while feed chamber #2 is loaded with the exposed and developed high contrast negative produced in the first pass. Both films are threaded through the camera together and attached to cores in the take-up chambers. By punching a hole in the film 80 frames (=5 feet for 35mm film) before the first picture, the animator can load the camera, close the cover, and advance both films exactly to the first frame. The colour areas are exposed as described before, but now the light from the CRT passes through the developed high contrast film before strik-

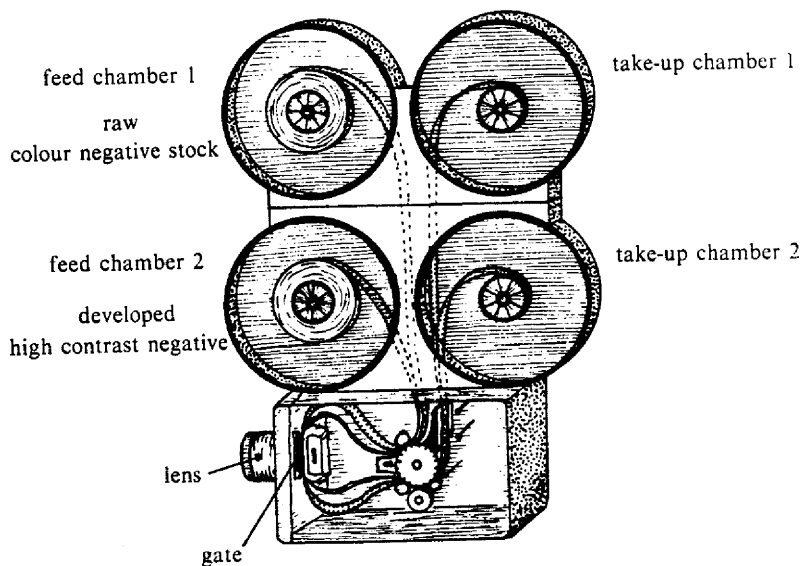


Figure 2.17 Camera with bi-pack magazine

ing the raw colour stock. Because the camera mechanism and the CRT are extremely precise, the black lines act as mattes, leaving thin borders around the colour surfaces unexposed. On the final positive colour print the resulting image consists of colour areas bordered by black outlines.

### 2.2.2.9. UTILITY

The UTILITY menu (Figure 2.18) provides a number of general housekeeping functions. Most of these are used to manipulate the various files generated by the other system processes. The animator first selects the file type (menu items #10-#16) and then the desired action, for example START PRINTING (menu item #8) to list a file on the printer, or DISK --> MAGTAPE (menu item #37) to create tape backups, and MAGTAPE --> DISK (menu item #35) to restore disk files from a tape backup. Another frequently used function is COMPRESS PIC FIL (menu item #22) which compresses the current picture file by recovering all space freed by deleting pictures and compacting that space into a single block. SWITCH PIC FILE (menu item #25) is used to select the current picture file. The system provides three picture files, but only one of them is open at any given time. PURGE PIC FILE (menu item #29) erases all data in the current picture file. This drastic action is only executed after the animator reconfirms his intention to do so. Finally, the UPDATE COLOUR function (menu item #2) allows the animator to change the entries in the colour lookup table which determine the intensities used for filming.

UTILITY	10 colour file	20	30 MOUNT MAGTAPE
1	11 pictures	21	31 DISMOUNT MAGTAPE
2 UPDATE COLOUR	12	22 COMPRESS PIC FIL	32 POSITION MAGTAPE
3	13 keyframe seq	23	33 LIST MAGTAPE
4	14 compose seq	24	34
5	15 skeleton seq	25 SWITCH PIC FILE	35 MAGTAPE --> DISK
6	16 preprocessed	26	36
7 ITEM/directory	17	27	37 DISK --> MAGTAPE
8 START PRINTING	18	28	38
9	19	29*PURGE PIC FILE	39 ERASE MAGTAPE
*****			
ARE YOU SURE YOU WANT TO PURGE PICTURE FILE #1 (Y/N) ? _			

Figure 2.18 The UTILITY menu

## 2.3. The Programmer's View of the Interpolation Process

The animation system described in the previous section is currently in use as a production tool at the National Film Board of Canada. Experience has shown that the interpolation process used in this system often gives the animation a mechanical look, usually referred to as the "computer signature". The remainder of this chapter examines the problems of computer assisted inbetweening. A number of previously published approaches are discussed, and a new algorithm based on Hermite interpolation is presented.

### 2.3.1. Problems in Computer Assisted Inbetweening

Computer animation has attracted researchers since the early days of computer graphics. It seemed that computers were ideally suited to taking over the repetitive task of generating thousands of fairly similar drawings to create animated motion. The initial enthusiasm has been somewhat dampened by the realization that the problem is not quite as easy as it had initially appeared. In fact, the development of a computer inbetweening system which can in all aspects match an experienced human inbetweener's skills requires solutions for a number of extremely difficult artificial intelligence problems. However, it is quite possible now to design a good computer *assisted* inbetweening system, provided that reasonable simplifying assumptions are made and that the animator can intervene when his automated assistant doesn't behave as expected.

An important factor in the success of such a system is the validity of the simplifying assumptions. Most inbetweening systems today use *linear keyframe interpolation*, an idea which was developed in some of the earliest animation systems. In this approach the simplifying assumption is that the path of motion of an object can be approximated by a sequence of straight line segments. Therefore, inbetweening can be performed by linearly interpolating between a set of key positions provided by the animator (Figure 2.19). Unfortunately, this straightforward approach has some undesirable side effects.

The most objectionable characteristic of linear keyframe animation is the jerky motion which makes characters appear like robots. These "clicks" are caused by discontinuities in the direction of motion at every keyframe. Unless successive key positions for a point are almost collinear (Figure 2.20a), the keyframes will be visible in the animation because there is a pronounced change in the direction of motion at every key (Figure 2.20b). The only way in which the animator can improve this situation is by increasing the number of keyframes (Figure 2.20c). Now there are more discontinuities, but each one is less severe. Unfortunately, the human eye is adept at perceiving even slight discontinuities, and as the number of keyframes is increased further, the economic benefits of computer inbetweening soon disappear. Thus simply adding more keys is not really a viable solution to the discontinuity problem.

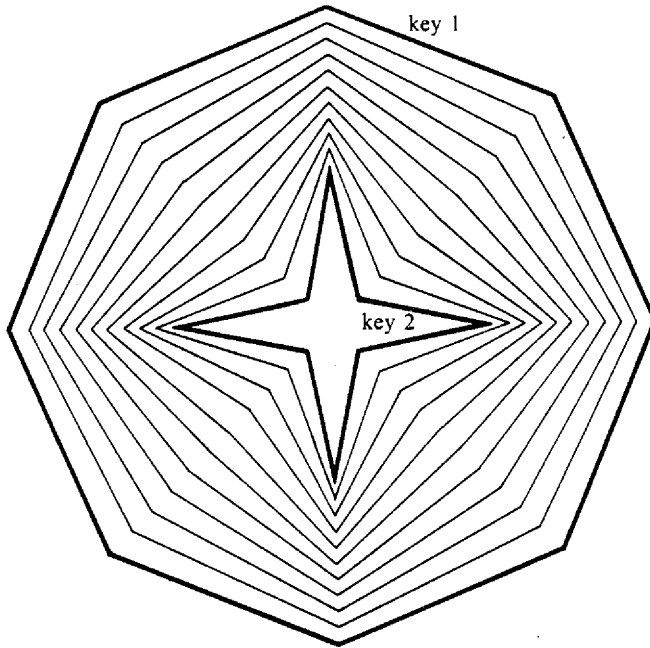
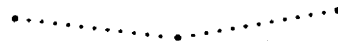


Figure 2.19 Linear interpolation between two keys

spe  
the  
chr  
sar  
bu  
tic  
tw  
th  
bo  
ke



(a) Minor discontinuity



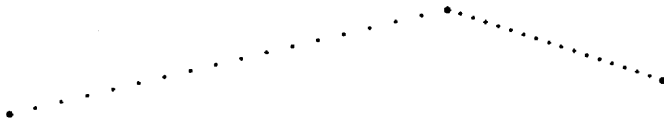
(b) Major discontinuity



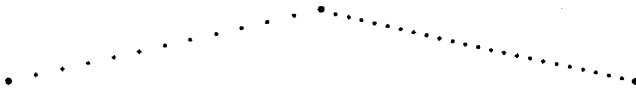
(c) Several minor discontinuities

Figure 2.20 Discontinuities in the direction of motion

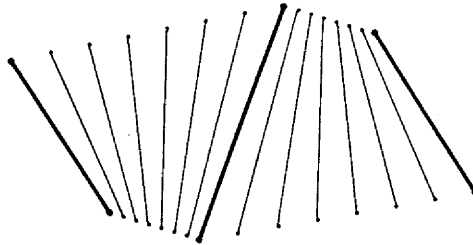
A related problem which is often not quite as obvious is discontinuity in the speed of motion. If the number of inbetweens in adjacent intervals is constant, but the distance between key positions is not (Figure 2.21a), the speed of motion must change suddenly at the keys because it is constant throughout any given interval. The same problem occurs when the distances between adjacent key positions are equal, but the numbers of inbetween frames per interval are not (Figure 2.21b). Acceleration and deceleration laws may be provided to allow the animator to smoothly join two parts of a motion. In practice this does not work well because different parts of the image may require different laws (Figure 2.21c). In most cases, discontinuities in both the direction and the speed of motion occur simultaneously, making the keyframes even more apparent.



(a) Equal number of inbetweens, but different distance between keys



(b) Equal distance between keys, but different number of inbetweens



(c) To smooth this motion, different interpolation laws would have to be applied to the top and bottom of the line.

Figure 2.21 Discontinuities in the speed of motion

A third common problem in linear interpolation of keyframes is distortion, which usually happens when there is a rotational component in the movement of an object. Linear interpolation finds the shortest path (straight line) from one position to the next, while rotational movements require a curvilinear path. The result is a shortening of the object which reaches its minimum length about half way between the keys. This effect is illustrated in Figure 2.22. Again the problem can be alleviated by increasing the number of keys, but this is not really a solution. [Burtnyk & Wein 76] proposed a method in which the animator explicitly specifies rotational components of movements. This technique works for cases such as the one shown in Figure 2.22, but in general it may be difficult to isolate rotational components in a complex movement which may, in addition, involve some degree of metamorphosis.

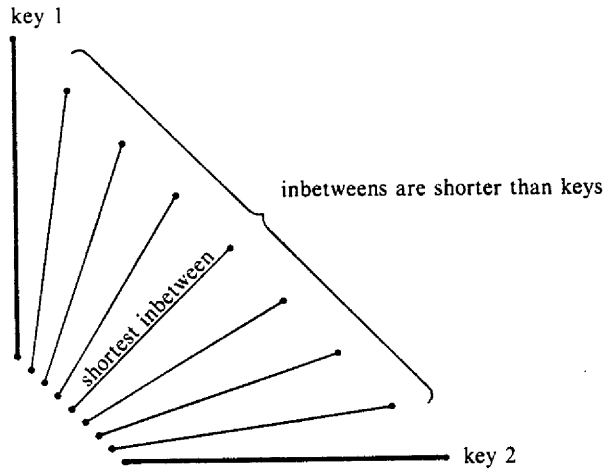


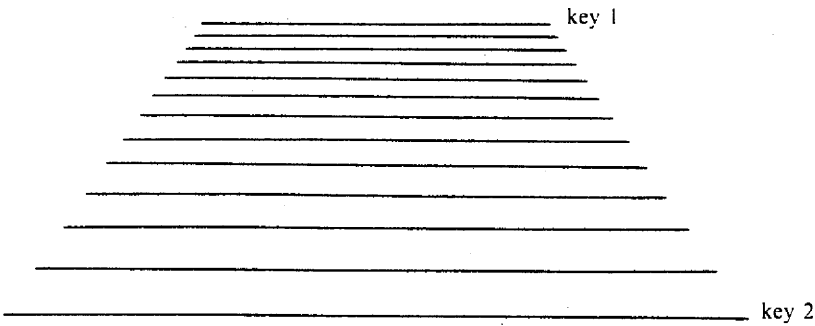
Figure 2.22 Distortion caused by rotation being simulated through linear interpolation

In 2-D images which are meant to give an illusion of 3-D depth, linear interpolation also has the undesirable property of visually flattening out any simulated movement in depth. The problem is that in reality an object moving towards an observer seems to move slowly at first, accelerating more and more the closer it gets to the observer (Figure 2.23a). Because this is the view *expected* by the observer, a strictly linear motion as shown in Figure 2.23b will actually appear to *decelerate* as the object approaches, and the perception of depth will be greatly reduced. To compensate the system could provide other interpolation laws such as accelerations for movements towards the observer and decelerations for movements away from him. Standard square law accelerations for which the increase in speed is constant are not adequate here, because motion appearing in perspective follows an *exponential law*. Exponential zooms have been used in traditional animation for a long time, but more generalized exponential interpolation laws are quite difficult to work with for animators, who are frequently not mathematically inclined.

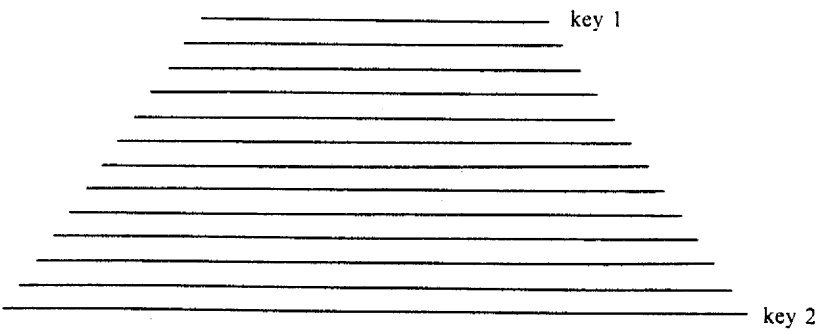
It would be preferable to use a linear interpolation law, but perform the calculations in 3-D space. This technique is useful even for a system oriented towards 2-D cel animation. Without going to full 3-D modelling, the image can be broken down into planes which are assigned to different depth levels. This approach is commonly known as "2-1/2D animation". By assigning actual z-values to each level and interpolating the z-coordinate before calculating the final perspective transformation, we can obtain realistic movement in depth (2-3/4D animation?) even with components which are completely flat drawings.



time  
draw  
anim  
  
trans  
the  
used  
2.24  
ing  
versi  
fore



(a) Exponential acceleration enhances illusion of depth.



(b) Linear interpolation destroys illusion of depth.

Figure 2.23 Movement in perspective

In view of the serious drawbacks of the straightforward linear interpolation technique, a number of different methods which produce smoother motion have been published. The next section describes four of these techniques which represent some very different approaches to the problem. One common characteristic of all these systems is that they are almost entirely *picture driven*.

obj  
tion

### 2.3.2. Previous Approaches to Computer Assisted Inbetweening

One of the first systems for computer assisted inbetweening was GENESYS, developed by Ron Baecker at M.I.T. [Baecker 69a,b,c]. In this system both images and movements are defined graphically. The animator first enters his drawings using a digitizing tablet. To control the animation he then traces out a motion path (*P-curve*) along which he wants the drawing to move. By sampling the tablet at regular

[B  
req  
a s  
car  
tra  
by  
(bc

time intervals, the system can determine not only the spatial characteristics of the drawn path (its shape), but also its temporal characteristics (the dynamics). Thus the animator simply mimics the desired motion, as shown in Figure 2.24.

A motion path by itself generates only some limited forms of animation based on translations. To allow the object to change shape while it is moving, in GENESYS the animator can also specify a *selection function* to determine which picture is to be used in any given frame along the P-curve. For example, for the path shown in Figure 2.24, the animator could make the impression of a bouncing ball much more convincing by specifying a selection function which replaces the round ball with a squashed version at the points of impact, and maybe with a slightly squashed version just before and after.

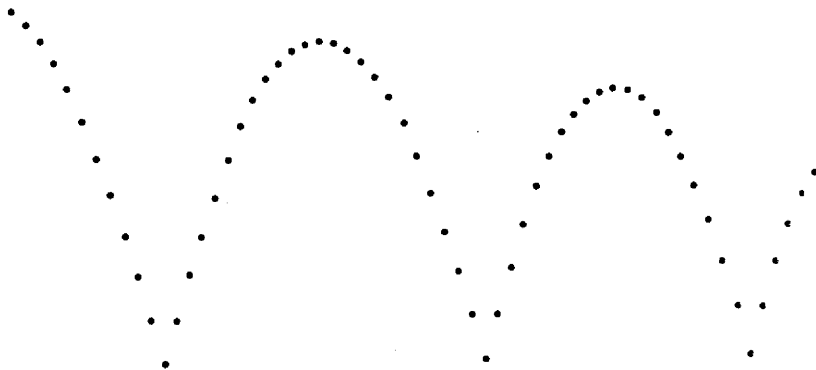


Figure 2.24 Motion path ("P-curve") for a bouncing ball

The P-curve technique gives the animator direct control over the motion of an object, but any effects requiring metamorphosis cannot be achieved with a single motion path.

The *skeleton technique* developed by Nestor Burtnyk and Marcell Wein [Burtnyk & Wein 76] was designed to handle the controlled metamorphosis which is required for character animation. As described in Section 2.2.2.4, the animator uses a simple stick figure skeleton to control the animation of more detailed drawings. He can manipulate the skeleton to construct key poses, and the system will automatically transform the detailed drawings correspondingly. This transformation is controlled by a set of quadrilaterals formed from the skeleton "bones" (centre lines) and "skin" (boundary lines), as shown in Figure 2.25.

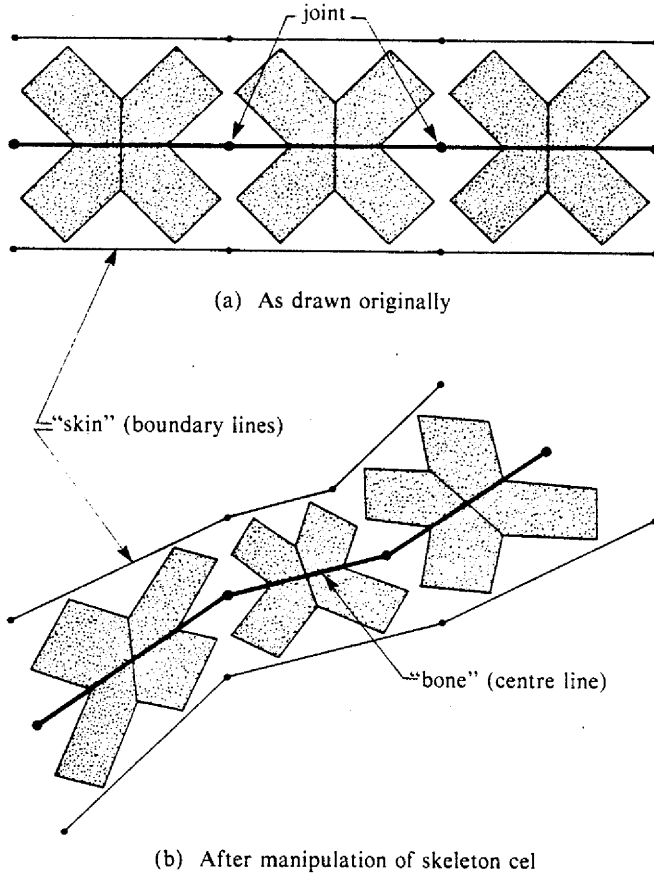


Figure 2.25 Skeleton cell and corresponding detail cell

Any manipulation of the skeleton will change the shape, proportion, or positions of these quadrilaterals (Figure 2.25b). Each point in the detailed drawing is converted to a relative coordinate system based on the vertex coordinates of the quadrilateral which encloses that point. Points which are not enclosed by any quadrilateral are not affected by the skeleton and will be interpolated linearly. To generate inbetween drawings, the system first interpolates the skeleton quadrilaterals, using piecewise cubic polynomials to obtain smooth motion. For each intermediate position of a quadrilateral, all points of the detailed image which were initially inside it are now reconverted to absolute coordinates using the vertices of the intermediate

quadrilateral as reference points. Thus the motion of the skeleton determines the animation of the detailed drawing of a character. Skeletons are a powerful technique because they closely simulate the animator's normal working procedure. To create a complex motion such as a walk, the animator usually eliminates all distracting detail and constructs the animation using a simple stick figure. Once the motion has been established, the stick figure is fleshed out and details are added.

To allow the animator to use skeleton control effectively, the system must provide a simple user interface which integrates skeletons well with the drawing phase and the inbetweening phase. At the same time the interface must be flexible enough to allow a wide variety of working patterns. Unfortunately, the system described in Section 2.2 does not have such a user interface, and therefore the skeleton process is only used infrequently.

The smooth cubic interpolation between keys eliminates clicks in movements controlled by skeletons, but clicks still occur in the animation *within* a skeleton quadrilateral. A common example is the use of skeletons to control the displacement and size of an object. To animate a bird in flight, the animator might prepare a few detailed drawings of the bird with wings up and down, as seen from several different angles. To move the bird along its flight path, the animator then constructs a simple box skeleton (a single bone with two boundaries). The flight path is defined by positioning, scaling, and rotating this skeleton box while the wing motion is produced by interpolating between the appropriate detailed cels. In the final animation the bird will follow a smooth flight path due to the cubic interpolation between skeleton positions, but the animation of the wings will exhibit the clicks which are so typical of linear interpolation.

A simple technique for avoiding clicks while still using a basically linear interpolation was proposed by Martin Tuori at the University of Toronto [Tuori 77]. In his system the animator can request that two different actions be overlapped. Thus the animator might specify that he wants to interpolate between Key A in frame 1 and Key B in frame 17, while also interpolating between Key A in frame 8 and Key C in frame 23. The result is illustrated in Figure 2.26. When frame 8 is calculated, the system first finds the current position of Key A by interpolating between the original Key A and Key B. This modified Key A is then used to execute the second action, interpolating between it and Key C. As a result the object begins to move towards position C before reaching position B. In effect the click which would normally occur at position B is avoided by rounding the corner in a shortcut. The major problem with this method is that some of the keys are never reached because the interpolation will bypass key positions which are overlapped with another interpolation which is not collinear.

Most recently, William Reeves at the University of Toronto has developed a technique for controlling keyframe animation through *moving point constraints* [Reeves 81]. His approach is a powerful generalization of Ron Baecker's P-curves which were described earlier. Instead of requiring a single motion curve for an object the system allows the animator to draw any number of curves for different parts of the object. Each curve defines the movement of some point in the image over a number of key positions. The number of moving points is determined by the anima-

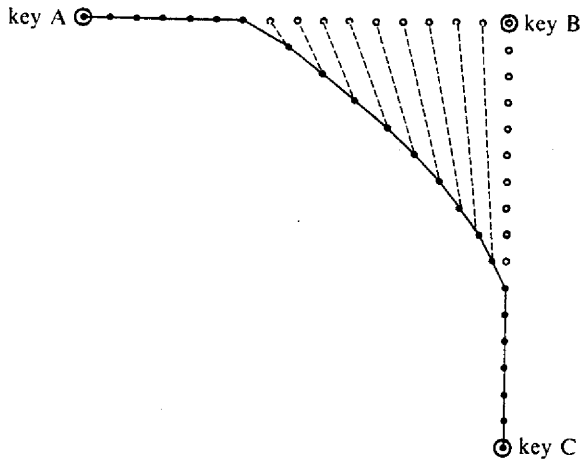


Figure 2.26 Overlapping actions

tor; in some cases a curve in a key may have no constrained points while another curve may have several. The curves in the key frames and the moving point paths together form a complete or partial patch network. The constrained points are interpolated according to this network while the other points are blended in, depending on the positions of the nearest constrained points.

The only drawback of this technique is that it becomes difficult for the animator to visualize the motion when there are more than a few moving point constraints, or to add constraints in such a way that the intermediate images preserve the structure of the keys. For example, to move a character's hand as shown in Figure 2.27, the animator could attach a moving point path to each fingertip. Unfortunately, even minor variations in the five paths can cause severe distortions in the intermediate frames. The above technique is probably most useful when the number of paths can be kept fairly small, for example when modifying only certain parts of a sequence by changing the current motion paths for some points.

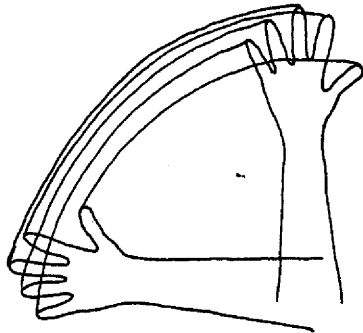


Figure 2.27 Moving point constraints

### 2.3.3. "Intelligent Inbetweening"

The techniques described in the previous section avoid the clicks of linear interpolation by allowing the animator to specify some information such as motion paths, skeletons, or action overlap in addition to the keyframes. Such an approach is analogous to a traditional animator having to give detailed instructions to his inbetweener who is not competent enough to interpret the animator's wishes when he is given only the key drawings. This section describes an approach which tackles the problem from a different angle by attempting to make the inbetweener more intelligent and independent.

#### 2.3.3.1. The Human Inbetweener

The basic job of a human inbetweener is to create intermediate drawings by interpolating between key drawings prepared by the animator. Without attempting to duplicate the inbetweener's expertise through artificial intelligence techniques, a look at his working habits can provide some insights which might be useful for automated inbetweening.

One fact which is immediately obvious is that the human inbetweener never starts drawing intermediate frames for an interval without first checking at least two key frames on either side. By contrast, a linear interpolation system simply looks at the immediately adjacent key on either side which defines the current interval. Thus the human inbetweener always works in a more *global context*. Knowing where an object will move once it has reached the current destination key allows him to prepare ahead of time by accelerating or decelerating the motion or by changing the path of movement to approach the destination key from a certain angle. Similarly the inbetweener must know from where the object arrived.

Most of these adjustments are based on an intuitive conceptual model of mass and inertia. The inbetweener knows that a moving object will only gradually change its direction or speed unless it is suddenly confronted by a major external force. This concept of mass and inertia is completely missing in a linear interpolation system. The resulting problems of discontinuity in the direction and speed of motion have been described in Section 2.3.1.

Another point which is quite intriguing is that many inbetweeners use a *recursive* strategy in their work. Given two keys at frames 0 and 8, they normally do not generate the intermediate frames 1-7 in that order, but instead start with the middle frame (#4). After adjusting this frame to their satisfaction, they are then left with two intervals with only three inbetweens which are again divided by first drawing the middle frames (#2 and #6). This "stepwise refinement" approach might also be useful for automated inbetweening systems. If the animator could "grab" an inbetween frame calculated by the system and modify it during the playback phase, the system could then treat this frame as a new intermediate key and re-interpolate the sequence accordingly. Thus a sequence might at first contain only a few keys, but their number could be increased where more control is needed by modifying inbetween frames, changing them into keys.

In view of these observations about human inbetweeners we can now define the characteristics we would like to see in an automated inbetweening system.

### 2.3.3.2. Desirable Properties for Automated Inbetweening Systems

A good inbetweening algorithm should have the following properties:

- predictability
- continuity
- flexibility
- conceptual simplicity
- local control
- computational simplicity

This section briefly discusses each of these properties. The interpolated frames should be fairly *predictable* from the key frames. This characteristic is crucial to allow the animator to prepare key drawings which will produce the desired results. An important consideration in this context is that the animator's keys usually represent extremes in the motion, therefore the algorithm should avoid generating too much overshoot or oscillation beyond the keys. While predictability is one of the most important properties, it alone is not sufficient, as demonstrated by linear interpolation which is perfectly predictable but fails in other respects.

*Continuity* is the other major factor determining the usefulness of an inbetweening algorithm. The problems caused by discontinuity in the direction and speed of motion have been discussed in Section 2.3.1. It is clear that positional continuity of the path of movement is not sufficient. Continuity of the first derivative (tangent) is a minimal requirement. The necessity for higher degrees of continuity has not been established yet. It should be emphasized that even though the problem could be stated mathematically as fitting a smooth curve through a set of sample points, factors in the human perceptual system should not be ignored. Given the graph of a curve, people can easily detect any major discontinuities in its second derivative (curvature) implying that continuity in the second derivative is a necessity. However, this may not be true for a curve which is not seen as a graph but as the path of movement of an object over time. Because only a fragment of the path is visible or in short term memory at any time, sensitivity to discontinuities in the second derivative may be reduced. It is hoped that tangent continuity will be sufficient if the curves themselves are well chosen.

The algorithm must be *flexible* enough to generate arbitrary paths, including multi-valued curves which are not restricted to a particular class such as conic sections. A mechanism should be provided to allow the animator to break the continuity of the curve in order to introduce cusps. At the same time the algorithm must be *conceptually simple*. The animator should not be asked to specify a large number of parameters to control the animation. The system should attempt to produce reasonable results given no more than a set of keyframes. If necessary it should be possible to modify this first interpolation further.

It is essential that the animator have *local control* when changing the animation. Any change in a single key should only affect that part of the sequence which is close to the modified key. The effect should be largest in the two immediately adjacent intervals and should not extend more than one or two intervals further.

*Computational simplicity* is also important because the algorithm may have to be applied to a large number of data points. This probably eliminates trigonometric functions as being too expensive. The storage requirements cannot be ignored either. Because a sequence may be arbitrarily long, the algorithm should work "on the fly" rather than requiring access to all keys in a sequence. Due to the large number of data points (up to 16,000 in a frame) storage overhead for working parameters must be kept as low as possible. This consideration is especially important for implementations on machines such as a PDP-11/34 with 256K bytes of memory.

### 2.3.3.3. An Algorithm for Smooth Keyframe Interpolation

The algorithm described here has all of the properties discussed in Section 2.3.3.2. It is based on the well known numerical analysis technique of *Hermite interpolation* [Ralston 65, Wendroff 69].

To generate intermediate frames we want to interpolate between a set of known sample points (the keys) using a simple smooth curve. Polynomials are a natural choice because of their simplicity, but interpolating polynomials of high degree tend to oscillate strongly beyond the sample points. Therefore, instead of using a single high degree polynomial interpolating all the key positions for a point, we want to construct the curve from pieces of lower degree polynomials, joined together with certain continuity constraints. The most suitable polynomials for this purpose are cubics because they do not oscillate much due to their low degree, but they are more flexible than quadratics because of the presence of inflection points.

Each of the cubic polynomials extends between two keyframes and is uniquely defined by four coefficients which we can determine by choosing four constraints. If we choose to have each of the cubics pass through the adjacent key positions with first and second derivative continuity at the keys, we obtain *splines* which are commonly used for surface modelling [Foley & VanDam 82]. A computationally simpler method results if we do not insist that the second derivative be continuous. Without second derivative continuity we can impose one other constraint. This freedom turns out to be very useful, because now we can not only require that the first derivative be continuous where the cubics are joined, but we can even impose a particular value for the derivative. Thus the constraints for each interval are the data values and the desired derivatives at the two adjacent keys. In practice we will choose "reasonable" values for the derivatives based on the geometry of the surrounding keys.

To make the algorithm sufficiently general for multi-valued curves, all functions will be treated parametrically, i.e. we will use  $x(t)$ ,  $y(t)$ ,  $z(t)$ . A separate cubic is found for each coordinate, but the calculations are completely symmetrical. In the following discussion, capital letters are used to represent triplets of coordinates, for example  $P(t)$  stands for  $P(x(t))$ ,  $P(y(t))$ ,  $P(z(t))$ .



Any cubic polynomial can be expressed as the sum of four basis functions, usually  $s^3$ ,  $s^2$ ,  $s$ , 1. However, for our purposes the four basis functions shown in Figure 2.28 are more useful. These functions have the following convenient property:

	$b_1$	$b_2$	$b_3$	$b_4$
function value at $s=0$	1	0	0	0
function value at $s=1$	0	1	0	0
derivative at $s=0$	0	0	1	0
derivative at $s=1$	0	0	0	1

Note that  $b_1$  alone determines the function value of the cubic at the start point of the interval. Therefore  $b_1$  must be weighted with a coefficient of  $F_0$ , the required function value at the start point of the interval. Similarly  $b_2$  must be weighted with  $F_1$ , the required function values at the end point of the interval. The derivative of the cubic is determined by  $b_3$  at the start point and by  $b_4$  at the end point, therefore  $D_0$  and  $D_1$ , the required derivatives at the interval end points, must be used as weighting factors for  $b_3$  and  $b_4$ . These observations lead to a cubic polynomial of the form:

$$C(s) = F_0 * b_1(s) + F_1 * b_2(s) + D_0 * b_3(s) + D_1 * b_4(s)$$

where  $s$  is a normalized parameter ranging from 0 to 1 over the interval.

To find appropriate values for  $D_0$  and  $D_1$ , we fit a parametric parabola through the desired point and one point on either side. Its slope at the point in question is used to define the slope of the two interpolating cubics which join at that point. To find this parabola, we first use the Euclidean distance between consecutive data points as a suitable parameterisation, i.e.

$$\begin{aligned} t_0 &= 0 \\ t_1 &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \\ t_2 &= \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2} \end{aligned}$$

Now the interpolating parabola in Newton form is:

$$P(t) = F[i+1] + F[i,i+1] * (t - t_{i+1}) + F[i-1,i,i+1] * (t - t_{i+1}) * (t - t_i)$$

where the square brackets denote divided differences [Conte & DeBoor 80]. Then the derivative at  $t_i$  with respect to  $t$  is given by

$$\frac{d}{dt}(P(t_i)) = F[i,i+1] + F[i-1,i,i+1] * (t_i - t_{i+1})$$

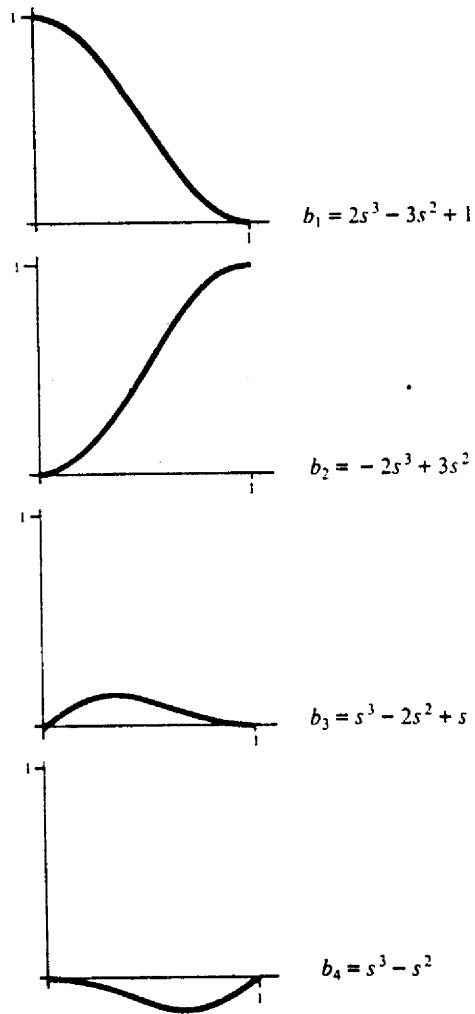


Figure 2.28 Basis functions for Hermite interpolation

To change from the original parameterization with respect to  $t$  to the normalized parameter  $s$  ranging from 0 to 1, we make use of the fact that  $s = \frac{t - t_i}{t_{i+1} - t_i}$ .

Because  $\frac{d}{dt} = \frac{d}{ds} * \frac{ds}{dt}$  we obtain

$$\frac{d}{ds} = (t_{i+1} - t_i) * \frac{d}{dt}$$

Thus with respect to the normalized parameter  $s$ , the derivative of the parabola is given by:

$$\frac{d}{ds}(P(t_i)) = (t_{i+1} - t_i) * (F[i, i + 1] + F[i - 1, i, i + 1] * (t_i - t_{i+1}))$$

A computationally efficient method of calculating this expression is given in Appendix C which contains program listings for two Pascal procedures which find the coefficients for the cubic interpolating polynomial and evaluate it to calculate an inbetween position.

### 3. WHAT NEXT?

Implementation of the curved interpolation algorithm described in Section 2.3.3 has progressed to the point where a short test film has been produced to demonstrate the feasibility of this method (Appendix D). This new interpolation process will be extended in the near future to include hidden surface processing and output of AREA data. At that time further development of the animation software will be frozen, and animators will continue to use it while development of a new system is started.

The current system has been useful as a prototype, but over the last two years it has become increasingly obvious that it lacks the power, reliability, and functional flexibility required for a production tool in a professional animation studio. Major changes in both the hardware and software are required, and it is no longer feasible to continue modifying the more than 500 Fortran and assembler routines in the system to make incremental changes.

The knowledge gained from the current prototype has led to a better understanding of the requirements for a production system, and plans are now crystalizing for the development of this next generation. Section 3.1 presents some preliminary thoughts about the design of the new system while Section 3.2 introduces the concept of temporal aliasing in motion pictures. This topic is currently an open research problem, but it has immediate application in the production of high quality computer animation.

#### 3.1. Towards the Next Generation

Experience has shown that the design effort for the next system must place much more emphasis on four important factors:

- ease of use
- reliability
- flexibility
- efficiency

The *ease of use* can be improved significantly by careful attention to good user interface design. More natural techniques must be employed for the animator's interaction with the system, drawing on experience with traditional animation wherever possible. The wide range of system functions must be structured much more carefully to provide easy access while eliminating the confusion of menus containing over 30 selections. Hardware changes in the animator's workstation can help to reduce the number of input and display devices to a more manageable level.

The system's *reliability* can be increased by selecting a hardware configuration with some degree of redundancy to allow continued operation in the event of equipment failures. Software reliability can be improved by adhering to strictly modular design and thorough unit debugging and testing. A more programmer-friendly environment than Fortran and RSX-11M (for example C and UNIX) would certainly help in this respect.

Another important consideration is that the system must be *flexible* enough to allow the animator to achieve the desired results. Since the current animation system was installed, the range of applications has spanned everything from 2-D character animation in the cel tradition to abstract geometric animation, titles, special effects, and technical 3-D animation. In many cases the system has been used in ways which the original designers had probably never anticipated. Obviously some of these projects have been extremely frustrating for the animators. It is therefore essential to provide a broader spectrum of functional capabilities and to avoid restricting the system to a particular style of animation.

The final requirement which deserves careful attention is *efficiency*. Changes in hardware are inevitable to allow animator input, interpolation, and filming to proceed in parallel. Efficient algorithms, especially for the interpolation process, are essential in view of the large number of frames which have to be calculated.

These factors are discussed in more detail below. Section 3.1.1 outlines a suitable hardware configuration for the new system while Section 3.1.2 presents some ideas concerning its software structure.

### 3.1.1. Proposed Hardware Configuration

The most important design consideration in the selection of new system hardware is the removal of current processing bottlenecks. To reliably support a high volume of production, a distributed system has several advantages over the present single processor system. A configuration with three or four workstations and a centralized processor (for example a VAX-11/780) should provide sufficient computational power to support the desired level of production. As shown in Figure 3.1, the central processor functions as a fileserver, managing mass disk storage as well as other shared resources such as tape drives and a line printer. Its main task is the interpolation process, which requires no animator intervention once it has been started. The filming station is controlled by a separate processor linked to the VAX for file transfers.

The workstations should be able to function independently of the rest of the network, if necessary. The components for such a workstation are shown in Figure 3.2. Computing power is supplied by a microprocessor, probably a Motorola 68000. Local working storage on Winchester disks allows most of the animator's interaction with the system to remain confined to the workstation. The picture manipulation wheels which are currently part of the animator's workstation have been removed to reduce the number of input devices. Their function can be easily simulated by appropriate tablet software, allowing the animator to use the tablet and stylus almost

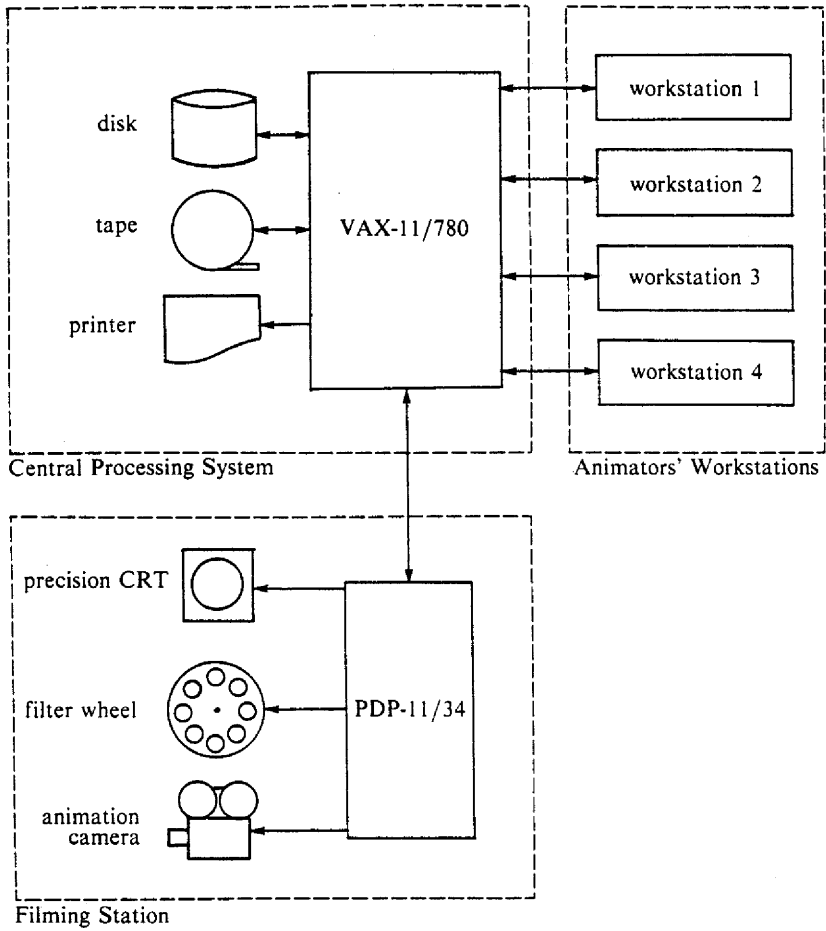


Figure 3.1 Proposed hardware configuration

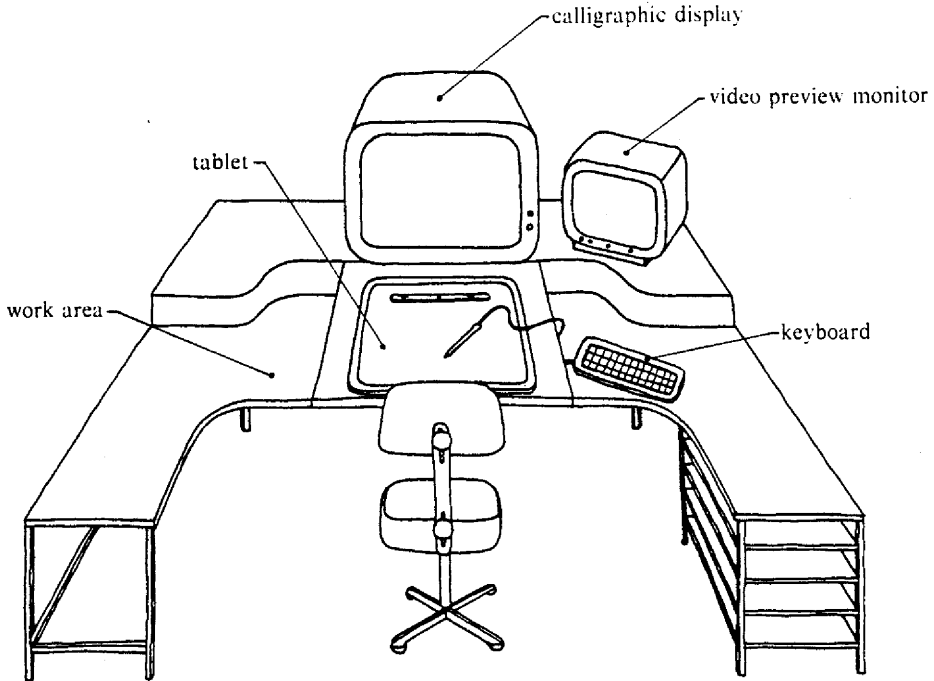


Figure 3.2 Proposed animator's workstation

exclusively. A potentially useful addition would be the new GTCO 4-D stylus which is advertised to provide tilt information in addition to  $(x,y)$  coordinates and proximity data. This device could be used to simulate a joystick without the inconvenience of an additional input device.

The keyboard has not been eliminated from the workstation. For certain tasks it is the fastest and most convenient input device. A detached keyboard would be best because it can be physically integrated into the workstation. The VT-52 alphanumeric display and the VT-11 graphics display have been eliminated in favour of a high performance calligraphic monitor which can display complex pictures and text menus simultaneously. The low resolution video monitor will probably still be needed for realtime playback of animated sequences. The workstation must include enough local storage for interpolating and previewing at least parts of a sequence without accessing the central processor.

By making the workstations as independent from the central processor as possible, certain system functions (e.g. input of drawings) will remain available even if the central processor fails. With three or four workstations there should be enough redundancy to reconfigure the system if necessary by switching components between stations, thus avoiding a complete shutdown in the event of hardware failures at the workstations.

### 3.1.2. Software Design

The design of the new animation system software must provide increased functional power while making the system easier to use for animators. The current two level menu structure (9 separate processes with up to 39 functions each) is unnatural and confusing. For the animator, drawing a picture and manipulating it are logically connected, and therefore the division into a DRAW and a MANIPULATE process seems artificial. Similarly, interpolating a sequence and previewing it should not require the execution of two different processes. The software design described below structures all system functions into five modules which correspond closely to the animator's conceptual model of the production process:

DRAW	- enter, manipulate, and edit drawings
SCRIPT	- construct sequences, compositions, camera effects etc.
INBETWEEN	- interpolate inbetween frames and preview
FILM	- output interpolated data to filming station
UTILITY	- provide miscellaneous housekeeping functions

The following sections outline the functional capabilities of these modules and their user interface.

#### 3.1.2.1. Menu Design

All menus are displayed on the calligraphic terminal to avoid shifting the animator's focus of attention to an auxiliary terminal. The number of functions which are required in each module makes it imperative that each menu be structured hierarchically rather than as a simple list of options. Logical blocks of selections such as transformations or edit functions are grouped together and turned on or off as a unit when necessary.

To reduce the clutter of complex menus the system displays only those options which are valid in the current context of operation. In addition every menu contains an ABORT selection to stop execution of the previous action as well as an UNDO function to reverse the last action and a HELP function to provide online documentation. A detailed discussion of good user interface design techniques is beyond the scope of this chapter, but the ideas presented in [Foley & Wallace 74] are a good starting point.



### 3.1.2.2. DRAW

The DRAW module provides all the necessary functions to enter cels into the system, and to manipulate, edit, store, and retrieve them. As described earlier a cel consists of a sequence of strokes, each of which contains a sequence of points. Logically this structure is quite similar to a text file consisting of a sequence of lines containing a sequence of characters. A fact which has not been recognized in the current system is that the functions required to input and edit drawings are not unlike those needed in a text editor. The current system does not provide any general picture editing capabilities, except the ability to add a stroke at the end of a picture or to erase the last stroke. A text editor with only these capabilities would be considered totally inadequate. What is needed is a graphical editor which is as powerful as most text editors. The animator must be able to select a stroke in the picture by tagging it with the stylus and to execute any of the following functions:

- APPEND - add one or more strokes after the selected stroke
- INSERT - add one or more strokes before the selected stroke
- DELETE - delete the selected stroke or set of strokes
- REPLACE - replace the selected stroke with one or more new strokes
- CHANGE - change some characteristic of the selected stroke (for example, change its type from OUTLINE to DETAIL)

In addition a RENUMBER STROKES function to change the order of strokes in a cel would be useful. It might also be helpful to allow the animator to REVERSE the order of points in a stroke, resulting in a stroke drawn in the opposite direction. The standard edit functions listed above together with the RENUMBER and REVERSE functions would easily correct most of the mistakes animators typically make when entering drawings into the system.

In addition to these stroke manipulation functions the system must provide more flexible graphic primitives. A major deficiency in the current system is the lack of text data. The new system will contain a number of predefined fonts as well as provisions for user defined fonts. The animator creates text by typing it on the keyboard and then positioning it interactively with the stylus. The system treats pictures containing text just like other pictures entered by the animator, allowing him to edit the strokes constituting a character or to change their ordering to produce any desired effect. Other desirable primitives include lines of different widths, coloured lines, and regular curve segments produced by an arc generator.

The functions currently found in the MANIPULATE process will be integrated into the DRAW module. The concept of using both digital and analogue input to specify transformations has been very successful and will definitely be incorporated into the new system. However, the usefulness of the picture manipulation wheels for analogue input is questionable. Their function can easily be simulated by tablet-based valuator [Evans et al. 81], thus eliminating the need to constantly switch

sequence 1

sequence  
dope sheet

be  
pr  
de  
fo  
sy  
W  
pa  
th  
ch  
er  
ce  
dr  
to  
str  
Fi  
se  
st  
sh

between input devices. The manipulation functions will also be extended to include a preview capability for the viewing transformations used in the new 3-D camera model described in Section 3.1.2.3.

Another area which requires careful consideration is the storage structure used for picture data and sequence control files. The three picture files used in the current system have no internal structure which could help the animator to organize the data. When a list of the picture file contents is requested, the system simply displays one page of picture names (see Figure 2.7) in the order in which they were entered into the file and allows the animator to go to the next or the previous page of names. This chronological ordering is quite useless because any cel which is updated to correct errors automatically shifts to the end of the file and is thus separated from related cels in the same part of the sequence. A single picture file may contain over hundred drawings, and the animator often wastes a lot of time paging through the picture file to locate cels.

Clearly some hierarchical structure is needed for picture storage. A tree structure similar to the UNIX file system is a natural model for animation data. Figure 3.3 illustrates how the data for a film can be organized in such a system. A separate *directory* is used for each composition. Under that directory the system stores the script control files for the entire composition, i.e. the composition dope sheet itself, skeleton files, camera effects, and transformations (see Section 3.1.2.3).

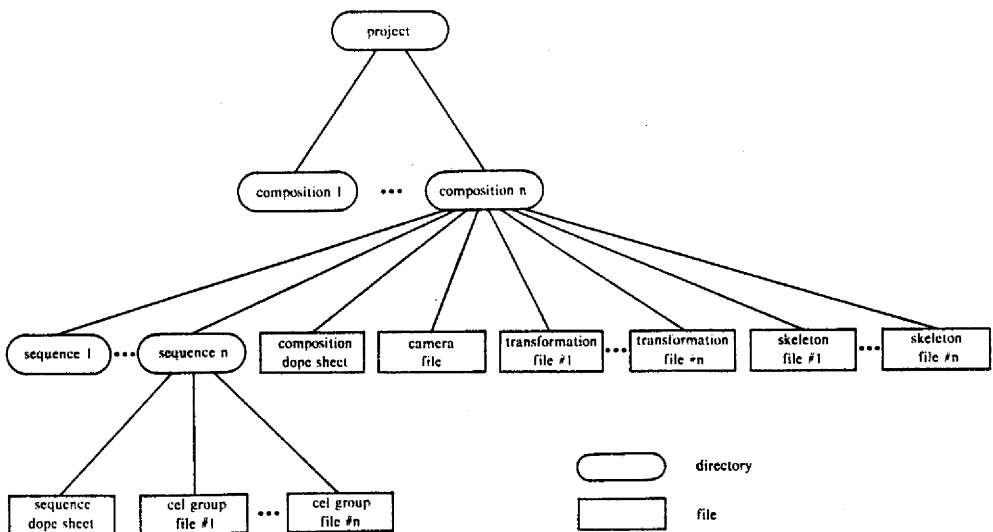


Figure 3.3 File structure for animation data

The picture data for the different sequences is stored under separate sequence subdirectories which also contain the *dope sheet* for the sequence. Within each subdirectory a number of *cel group* files are used to store collections of cels which are logically related, for example all the different head positions for a character within a sequence.

All data for active animation projects as well as other system files are stored in a centralized file system managed by the VAX-11/780. At the beginning of a session the animator downloads the part of the file system he will be using into the local storage of the workstation. At the end of the session the contents of the local storage are copied back to the main file system. The centralized file system can easily be backed up on tape at regular intervals.

### 3.1.2.3. SCRIPT

The SCRIPT module contains all the functions which are required to define the action in a composition once the key drawings have been entered. In this module the animator can create and modify

- sequence files
- composition files
- skeleton files
- camera files
- transformation files

One major problem in the current system is the design of the GENERATE process. The table format which is used for entering sequence information (Figure 2.10) is cumbersome and unnatural for the animator.

Traditionally animators are used to specifying the action in a sequence by using a *dope sheet*. This sheet contains one column for each cel level in the sequence (for example a body column, head column, legs column, and background column) and one line for each frame. Camera effects and other special instructions are indicated on the sheet with graphic codes (arrows, lines, etc.). This layout is easy to follow because the information is presented in a clear two-dimensional graphic format. The separate columns allow the animator to independently specify actions for different parts of the picture.

In the current keyframe table it is possible to enter several cels for a keyframe, but it is difficult to animate them independently. For example, if a character changes its head position frequently, the animator may need to define a keyframe at every fifth frame. Unfortunately he then also has to specify all the other cels for that frame (legs, body and background), even though they could probably be animated with far fewer keys.

This problem can be overcome by entering the sequence file in a form similar to a *dope sheet* (Figure 3.4). The leftmost column contains frame numbers while the remaining columns are used to specify the cels for up to six independent *actions*. The

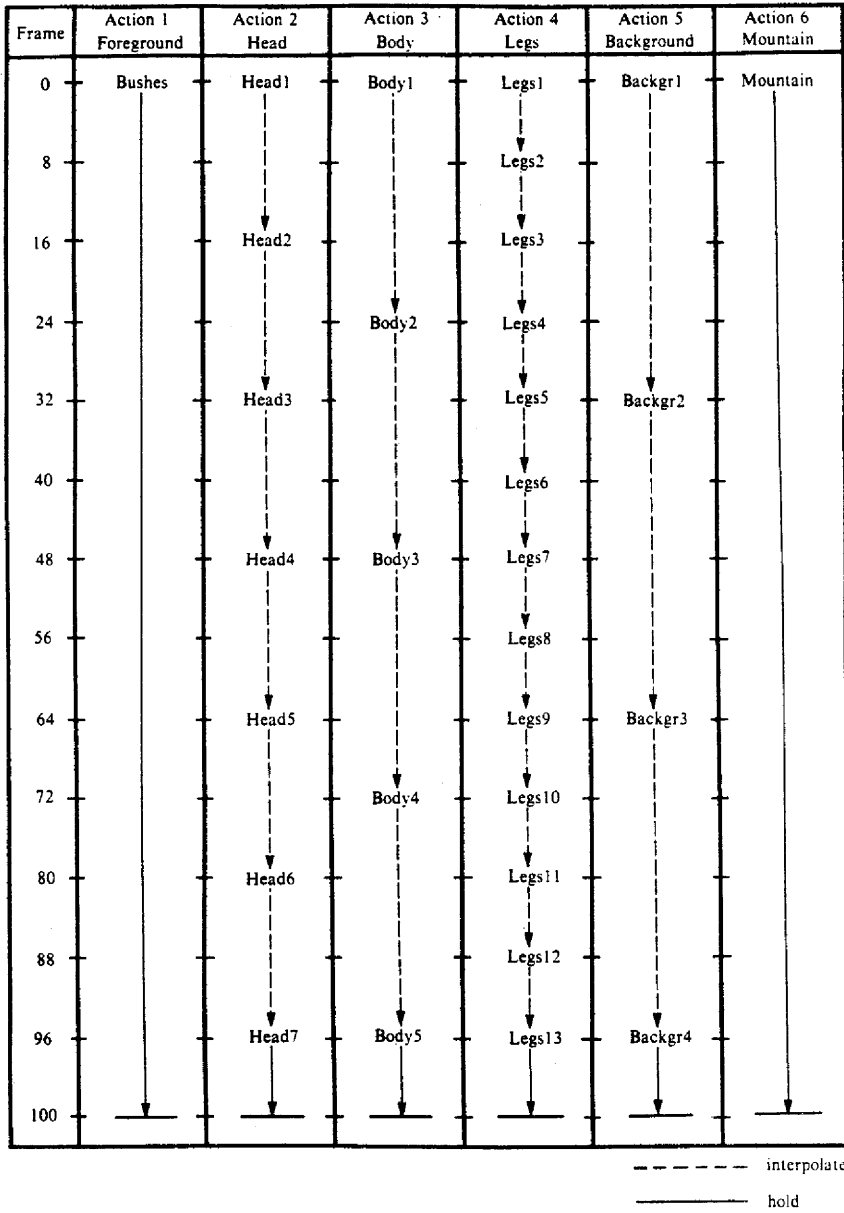


Figure 3.4 Sequence dope sheet

dope sheet is displayed on the calligraphic terminal, and the tablet and stylus are used to select lines or columns and to enter symbols. Again the functions required in this module are similar to those used in a text editor. The animator must be able to add, delete, replace, move or copy lines in the table. In addition special functions to automatically generate cel *cycles* would be useful. Cel names can be typed in on the keyboard or selected from a list displayed on the terminal. Whenever the animator specifies a cel name, a reduced version of the drawing is shown in one area of the display screen to allow the animator to verify his selection. All information entered by the animator is checked for consistency (for example, frame numbers must be strictly increasing) before the data is written into a sequence file.

Once a consistent sequence has been entered, the animator can request a line test of any one of the six actions. This line test is produced by interpolating the action "on the fly", using curved interpolation without hidden line removal, and displaying the result on the calligraphic monitor. The playback rate is normally slower than 24 frames per second, but the line test allows the animator to catch major flaws in the animation such as stroke ordering errors or misnamed cels. In addition the animator can "grab" any of the inbetween frames produced during the interpolation, then modify and store it just like any key drawing. A facility for specifying moving point constraints [Reeves 81] should also be provided to allow the animator to change parts of the animation if necessary.

The SCRIPT module also contains functions which allow the animator to edit a composition file. As shown in Figure 3.5, a composition consists of references to a number of sequence dope sheets, *skeleton files*, *transformation files*, and a *camera file*. The skeleton files are similar to skeleton sequences in the current system, but

		0	80	200	360
Man1	SEQ	----->			
Man2	SEQ	----->			
\$Man2	SKL	----->			
Car	SEQ	----->			
Carmove	TRN	----->			
Ball1	SEQ	----->			
Ball2	SEQ	----->			
\$Ball2	SKL	----->			
Ball2move	TRN	----->			
Dog	SEQ	----->			
\$Dog	SKL	----->			
Bird	SEQ	----->			
Birdmove	TRN	----->			

Figure 3.5 Composition dope sheet

the information is entered in the form of a dope sheet rather than a table. Transformation files are used to specify dynamic translations, rotations, and scaling which are to be applied globally to a sequence. The animator can create a different transformation file to control each sequence in the composition. In addition he can specify another transformation file which affects the entire composition. The camera file is used to define the 3-D viewing parameters for the composition. The animator can select a series of camera positions, viewing directions, and focal lengths, and the system creates smooth camera movements by applying a curved path interpolation to these parameters. Figure 3.6 shows the hierarchy of transformations which are applied to points in the original drawings to produce the final inbetween frames.

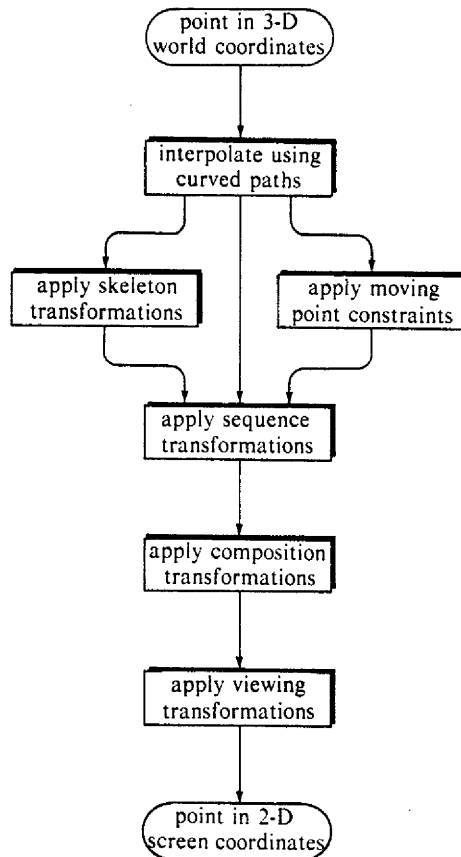


Figure 3.6 Hierarchy of transformations

### 3.1.2.4. INBETWEEN

The INBETWEEN module contains the functions currently found in the INTERPOLATE and VIDEO processes. The calculation of inbetween frames is based on the curved interpolation described in Section 2.3.3 and the other transformations shown in Figure 3.6. The interpolation is the most computation intensive part of the system and will usually be run on the VAX. The output files generated by the interpolation process are stored on disk in the central file system and can be sent to the workstation for previewing, to the filming station for hardcopy output, or to tape for archiving. A major gain in speed compared to the current system can be made by having the system interpolate both HIDDEN LINES and AREA data simultaneously, writing the data to two different disk files. Hidden surface processing is one of the most time consuming aspects of the interpolation phase, and the current animation system must perform these calculations twice because two separate interpolations are required if both LINE and AREA data is needed. The hidden surface algorithm in the new system will be based on actual z-depth, and the animator therefore does not need to manually define depth levels for each cel in the image.

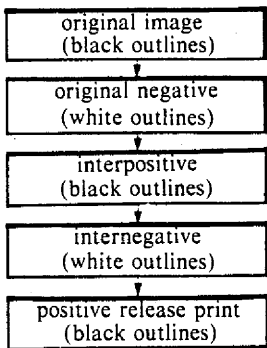
The workstation must at least be able to interpolate compositions in LINES and HIDDEN LINES mode without accessing the VAX, if necessary. However, this capability is limited by the amount of available local disk storage. Processing for video preview is done entirely within the workstation. The preview function is coupled with the interpolation so that the animator does not have to execute each function individually. If he requests a preview for a composition, the system simply checks whether an up-to-date interpolation file for the composition exists, and if necessary performs the interpolation before executing the preview program.

### 3.1.2.5. FILM

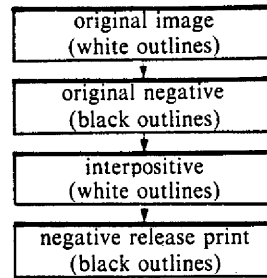
The FILM module allows the animator to submit interpolated sequences for hardcopy output. Because filming is treated as an independent offline process, the FILM module in the workstations is merely used to specify jobs for the filming station and forward them to the central processor. The VAX dearchives the requested file from tape if necessary and, as soon as the filming station is available, starts transferring the file to the filming station processor.

If preliminary tests are successful, the filming process will use a new approach to combine line and area data, eliminating the second pass required by the bi-pack technique employed in the current system. As noted in Section 2.2.3.8 the principal problem is that any part of the film which has been exposed to light cannot be changed back to black. However, it is quite possible to make any area on the film appear white by exposing it to more light. This technique, often referred to as *burn-through*, is commonly used to superimpose credits or subtitles over a sequence. Thus we can easily obtain white lines. We can convert these into black lines fairly simply by using a negative print of the film. Unfortunately all other colours in the image are also reversed to their equivalent negative colours. To compensate for this effect we can photograph the entire sequence by working in a negative colour space.

Figure 3.7 illustrates this idea. To produce a final image of a *red* circle with a *black* outline, we instead film a *cyan* circle with a *white* outline. On the negative these colours will appear as red and black. Now instead of making a normal positive print which would show exactly the colours we photographed, we make a negative print which preserves the red and black colours from the original negative. Thus lines are simply burnt through on top of the colour areas in a single pass. The entire process can be completely transparent to the animator if the system performs the mapping from positive to negative colour space automatically. A question which cannot be answered until this method is tested is whether a satisfactory range of colour can be obtained by working in a negative colour space.



(a) Normal printing procedure



(b) Printing procedure to obtain negative image

Figure 3.7 Burn-through on colour negative film

### 3.1.2.6. UTILITY

The UTILITY module provides miscellaneous housekeeping functions similar to those found in the current system. This menu allows the animator to transfer files and directories between the central file system and the workstation and to request tape back-ups to be taken for disk files. Other functions provide access to the central printer to produce listings of picture files, dope sheets and other textual information.

The ideas presented so far in this chapter constitute a logical framework for the design of the next generation of the animation system. While much more work is needed to refine these concepts, the requirements and possible solutions are beginning to be fairly well understood. The next section discusses the problem of temporal aliasing in motion pictures. It is becoming increasingly clear that temporal antialiasing techniques could make a significant contribution to an improved quality of motion in computer animation. At the present time no such algorithms have been published, and the next section is only intended to give an introduction to the topic. If research in this area produces practical results in the near future, temporal antialiasing algorithms will probably be incorporated into the interpolation and filming programs for the new animation system.



### 3.2. Temporal Aliasing in Motion Pictures

Even when the techniques described above are applied to generate smooth motion along curvilinear paths, the final result on film is not always satisfactory. The problem is that motion pictures are exactly what the name implies - still pictures of motion. By presenting a sequence of these still pictures sufficiently quickly, the human eye and brain can be led to believe that the image is actually moving. Thus motion pictures approximate a continuous event (smooth motion) by a set of discrete samples (24 still frames per second). Conceptually this is very similar to the idea of representing continuous tone images by a set of discrete samples (pixels) in a frame buffer. Therefore it is not very surprising that the spatial aliasing problems (jaggies, Moire patterns etc.) commonly found in raster graphics have temporal counterparts in motion pictures.

A common example of temporal aliasing occurs when an object moves across the screen too quickly. Beyond a certain speed threshold the eye can no longer fuse the individual pictures into a smooth motion; the object seems to skip across the screen in discrete jumps. This phenomenon, which animators call *strobing*, is equivalent in the spatial domain to the "jaggies" in raster graphics. Both strobing and jaggies are caused by the fact that the eye is very sensitive to discontinuities and, beyond some threshold, will perceive a set of discrete samples rather than the continuum which the samples represent.

Another example of temporal aliasing is the "wagon wheel phenomenon" often seen in Westerns. A stagecoach, pursued by a gang of bandits, thunders across the screen at full speed when suddenly its wheels seem to stall and even turn slowly backwards. This is a stroboscopic effect which can also be produced by illuminating a rotating object with a strobe light source. When the scene was shot the wheels were rotating at a critical frequency which could not be captured in 24 exposures per second. The camera sampled the continuous motion in such a way that the high frequency forward rotation was *aliased* as a low frequency backward rotation (see Figure 3.8).

Aliasing is an important problem in raster graphics. The discrete sampling used to produce an image introduces artifacts such as jagged edges and Moire effects which are analogous to the wagon wheel phenomenon in the temporal domain. So far most of the effects which have been studied are spatial phenomena, but some temporal effects such as small objects "dropping out" between the scan lines or "escalators" (moving jaggies) have also been observed in animated raster images.

To study the phenomenon of temporal aliasing we must first have a basic understanding of the mechanics underlying the production and perception of motion pictures. The next sections introduce some simplified but fairly realistic models of motion picture cameras and projectors. The remainder of this section discusses temporal aliasing in animation and some approaches for solving the problem in computer animation.



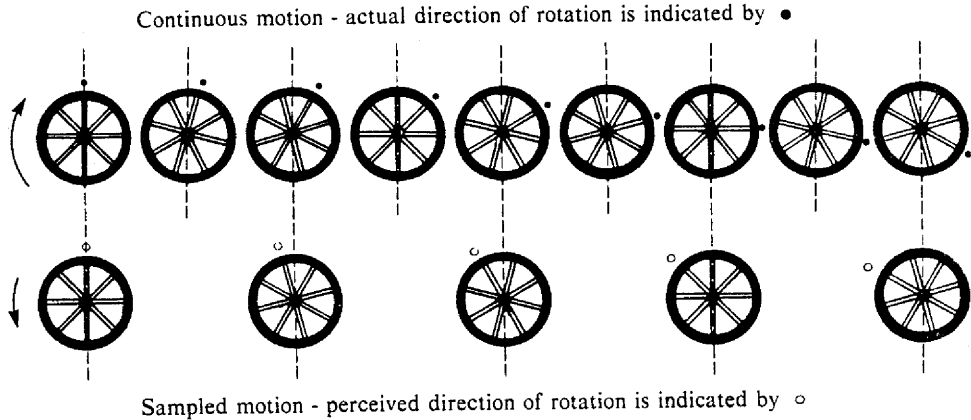


Figure 3.8 The “wagon wheel phenomenon”

### 3.2.1. Motion Picture Cameras

As mentioned earlier, a standard motion picture camera records 24 frames of film per second. This is achieved through the use of an intermittent film movement and a synchronized shutter. Twenty-four times each second the shutter is closed to block out incoming light, the film is advanced by the intermittent movement, and as soon as the film motion stops the shutter opens to expose one frame. Most 16mm, 35mm, and 65mm cameras employ a *rotary shutter* which is simply a spinning disk with a cut-away sector. One revolution of the disk corresponds to the positioning and exposure of one frame of film and takes  $1/24$  of a second. The percentage of the shutter which has been cut away is determined by the speed of the camera's film advance mechanism and varies between manufacturers. Most cameras have an open segment equivalent to  $175^\circ$ , an almost semi-circular shutter blade giving an effective exposure time of  $1/50$  of a second. The 65mm Panavision and 35mm Panaflex cameras have a very fast film advance mechanism allowing a  $200^\circ$  shutter and an effective exposure time of  $1/43$  second.

Figure 3.9 illustrates one cycle of a  $180^\circ$  shutter. The “wasted angle” indicated in the diagram is the shutter position range for which the camera aperture is neither fully open nor fully closed. By moving the aperture away from the shutter rotation centre this angle can be kept to a few degrees and therefore a square wave can be realistically used to model the light input function.

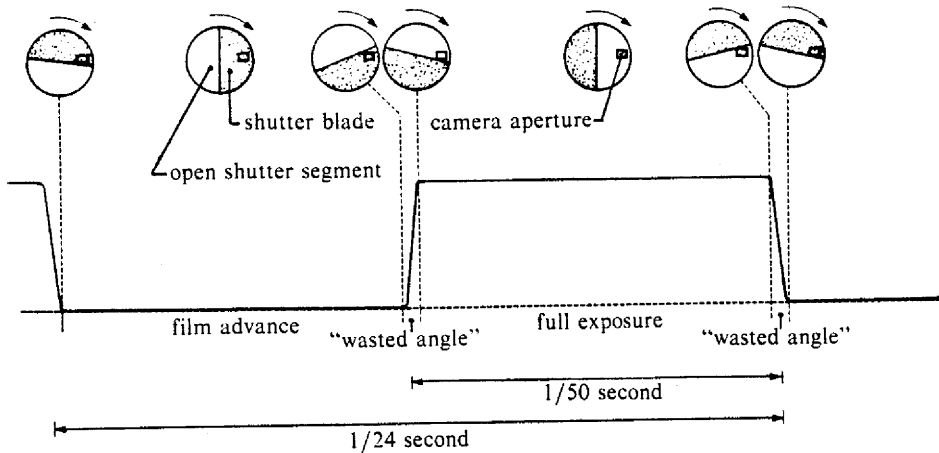


Figure 3.9 Light input function for a 180° rotary camera shutter

### 3.2.2. Motion Picture Projectors

Mechanically projectors are quite similar to motion picture cameras. An intermittent movement is used to pull down the film, and a rotary shutter cuts off the projector light while the film is moving. However, the design of a projector shutter is quite different from that of a camera shutter.

While a convincing illusion of motion can be generated with as few as 12 different frames per second, a film projected at this rate would exhibit unacceptable flickering due to the alternation of dark and light periods caused by the light cutoff during the film advance phase. If the rate of flicker is high enough the eye does not perceive the flicker, but sees a continuous image. For humans this *critical flicker frequency (CFF)* lies around 45 Hertz; its exact value depends on environmental factors such as the screen brightness level and image contrast.

Silent films were photographed at a rate of only 16 frames per second, but as early as 1903 a projector had been developed which operated above the critical flicker frequency. The same design is still in use in silent film projectors today. The projector uses a rotary disk shutter with three open sections and three blades. When the first blade (the *working blade*) blocks the projector light beam, the film is advanced and the passing of the remaining blades (the *balancing blades*) and the open segments causes the frame to be flashed onto the screen three times. One complete rotation of the shutter takes 1/16 of a second. Thus even though only 16 different frames are projected each second, projecting every frame three times raises the flicker frequency to 48 Hertz and allows the human eye to fuse the flashes into a continuous image.

When sound was introduced in the film industry, the 16 frames per second filming rate was no longer adequate because it did not provide enough bandwidth for adequate sound reproduction. The resolution of optical soundtracks was not sufficiently good to record high frequency sounds, given the fairly low speed at which the film passed the optical sound reader in the projector. The standard filming rate was increased to 24 frames per second and sound projector shutters were designed to have only two blades, one working blade and one balancing blade. Each frame is thus projected twice, and the effective flicker rate is again 48 Hertz. The two blade shutter has a better total light transmission (50-55%) than the three blade shutter (40-45%) because the working blade for a three blade shutter is larger than the balancing blades, leaving less than half the shutter area open. Figure 3.10 illustrates the result of one complete revolution of a two blade shutter. Note that the film is not moved when the balancing blade cuts off the projector light. Thus the only function of the balancing blade is to increase the flicker frequency to 48 Hertz.

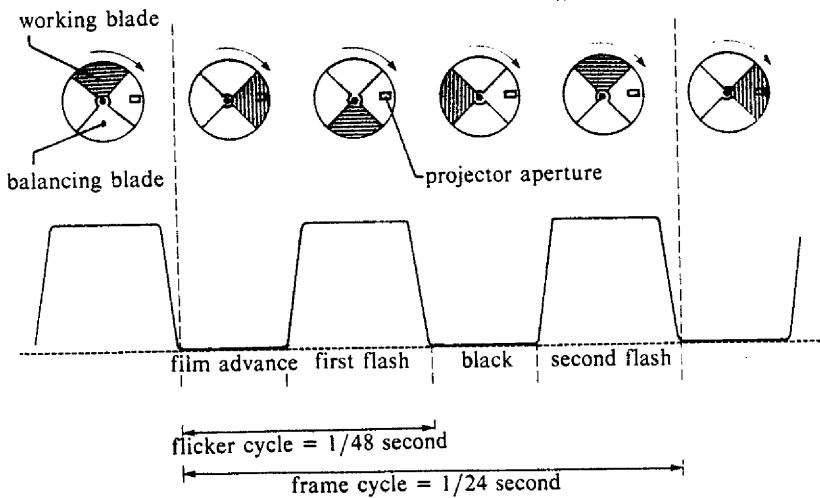


Figure 3.10 Light output function for a two blade projector shutter

### 3.2.3. Temporal Anti-Aliasing

The occurrence of temporal aliasing effects is influenced by three factors:

- the camera
- the projector
- the human eye and brain

The mechanisms in the human perceptual system which allow us to detect and interpret motion are not fully understood yet, and an in depth discussion is certainly beyond the scope of this thesis. The analysis presented here is based on empirical knowledge of the perception of motion in films. No attempt is made to define a precise psychophysical model for the human perceptual mechanisms which detect motion.

For practical purposes the projector can be treated as a known constant. Lance Williams suggested in [Kovacs et al. 82] that the multiple projection of each frame may be a contributing factor in our perception of strobing. The eye is probably tracking the motion of an object when it is suddenly presented with a second flash of the object in the same position rather than in the predicted new position. This could lead to a visual doubling of edges in the moving object. While this effect may contribute to strobing, there is not much that can be done about it given the 24 frames per second motion picture standard and the required 48 Hertz projection flicker rate.

The most promising area of research is how the motion picture camera should be modeled in the production of computer generated animation. Temporal aliasing is clearly a result of representing high frequency motion by a low frequency set of still frames. In video production the sample frequency could be doubled if odd and even fields were calculated separately, but for film this is not possible due to the established projection standards. To obtain a satisfactory result at the low sample frequency the high frequencies must be filtered by applying some *averaging techniques*. In effect we want to blur the individual frames slightly so that they represent a period of time rather than an instant of time. In raster graphics this is equivalent to sampling and averaging intensities over a pixel area. In video recording secondary factors such as NTSC encoding and phosphor persistence blur the image automatically. Therefore the following discussion is largely restricted to film.

When filming live action this temporal low pass filtering actually occurs automatically because the camera shutter stays open for typically  $1/50$  of a second. Thus a moving object is slightly blurred as a single frame captures almost half of the movement between two consecutive frames. By contrast, animation is produced by individually photographing a sequence of still drawings or static objects. Thus a frame of animation represents a sharp "snapshot" of an instance in time, not the frame of averaged movement seen by a live action film camera.

This fundamental difference explains why strobing is so much more common in animation than in live action. At sufficiently high speed even motion filmed from live action will begin to strobe even though there is always some blurring. An interesting

comment in this regard is provided in the *American Cinematographer Manual* [Clarke & Tyler 80] comparing the Panavision 200° camera shutter (exposure time = 1/43 second) with the standard 175° shutter (exposure time = 1/50 second):

*"In addition to allowing for the use of smaller lens stops, the 200° shutter significantly reduces the 'skipping' effect of fast pans or rapidly moving subjects."*

Strobing has been a known problem in traditional animation for a long time. To avoid it animators usually follow some empirical rules of thumb such as these:

- Don't move objects more than 1/30 of the screen width in a single frame.
- Avoid high contrast because it aggravates strobing.
- Avoid thin lines and repetitive patterns. They strobe more easily than thicker lines and irregular patterns.
- Don't move an object more than half of its own width in a single frame.

These informal guidelines can probably be related directly to the Nyquist theorem of signal processing, itself the basis for much of the work on spatial antialiasing in computer graphics. The last rule is especially interesting. Phrased slightly differently it implies that the position of an object in a given frame should at least partially overlap with its position in the previous frame. This will make it easier for the eye to fuse the two frames into a motion instead of perceiving them as two separate positions. Of course in some situations dramatic necessity requires that all of these rules be broken. If a high contrast object must be moved quickly, animators usually provide some artificial averaging by drawing *speed lines*, also known as *motion blur*, to blend between successive frames.

Figure 3.11 illustrates what happens when continuous motion is filmed in live action with a 200° and a 175° shutter. It also shows how the motion would appear if it were animated with and without motion blur. Potential strobing problems for this motion can be predicted from looking at the relative continuity between successive frames. With a 200° shutter the position ranges recorded in adjacent frames clearly overlap and thus strobing would not be a problem. With a 175° shutter the position ranges touch and overlap only slightly, but still strobing should not be a major problem. However, the animation without speed lines would definitely produce strobing since successive positions do not even touch. Adding artificial speed lines would significantly reduce this problem.

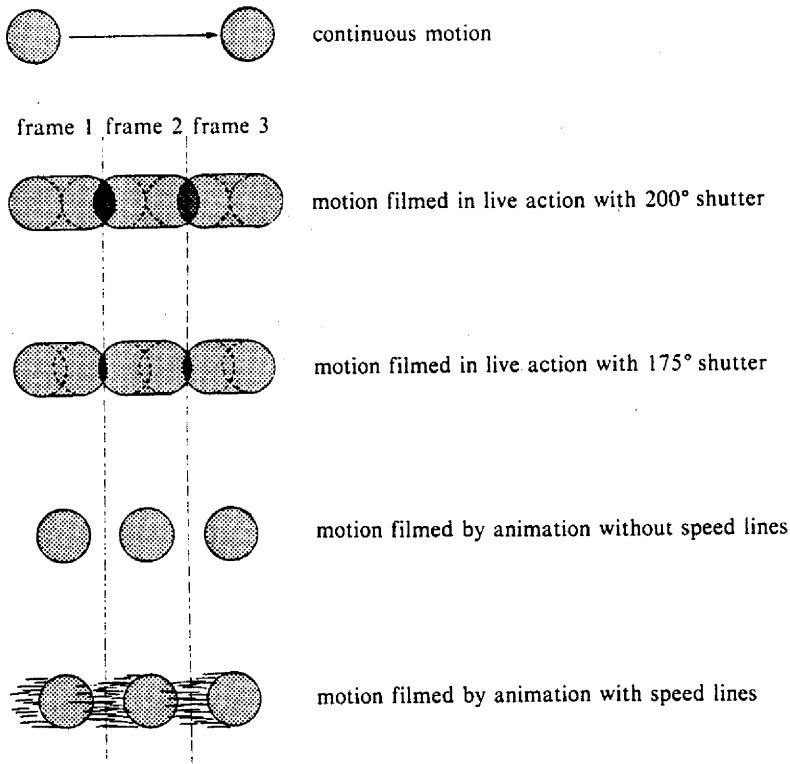


Figure 3.11 Blurring of motion in live action and animation

### 3.2.4. Solution Approaches in Computer Animation

One advantage of computer animation is that it offers a much more controllable environment than conventional animation. The generation of motion blur is quite difficult in manually produced animation, and at best a very coarse approximation is used. However, the incorporation of *automatic* motion blur into a computer animation system does not present any major conceptual difficulties since a mathematical model based on "natural" motion blur from a live action camera can be used. Ideally the individual frames should be blurred just enough to eliminate strobing without giving a "fuzzy" appearance to the sequence.

In a keyframe animation system the generation of frames is controlled mathematically, and thus the system could determine intensity values for each frame by integrating the intensities contributed by each object during its movement from the previous frame. This is logically equivalent to leaving the camera shutter open for a short time period while the objects are moving. Integrating over the entire time period between frames (1/24 of a second) would probably blur the image too much, but integrating over half of the interval should work since this is exactly what a live action camera with a 180° shutter does. In view of the characteristics of 200° shutters described earlier, integrating over more than half the frame interval may reduce strobing even further without producing a noticeable degree of fuzziness in the sequence.

Modeling the camera mathematically would allow us to further experiment by setting parameters such as the exposure time (i.e. the integration interval) to values which would be physically impossible in live action cameras. In a production system, efficient approximation techniques would have to be used to integrate the intensities over the desired time interval. Initially it might be best to ignore computational overhead and simply compute some number of *sub-frames* distributed over the desired integration interval. These sub-frames could then be averaged digitally in a frame buffer. Alternatively an analogue technique which might be useful would be to superimpose the sub-frames directly by making multiple reduced intensity exposures on a single frame of film.

Since human perception is not sufficiently well understood to allow the construction of precise models, ultimately the success of these ideas must be judged by empirical evidence. Using different averaging intervals, motion speeds and picture contrast values a set of test sequences should be filmed to determine the parameter values which are good for human viewers. Even though some insight may be gained by using frame buffer animation techniques and video tapes, the final output must be on film. Due to phosphor persistence, interlace scanning and different cycle rates, results obtained for video are not necessarily applicable to film. Moreover, strobing is much more severe when viewed on a large projection screen because the speed of motion relative to the observer (angular displacement within the field of vision) is much greater than if the same sequence were viewed on a typical home television set.

An understanding of the mechanisms described in this chapter and insights gained from the extensive research on spatial antialiasing should be combined to develop efficient and effective temporal antialiasing techniques for computer animation.



## **APPENDIX A**

### Glossary

**action**

An action is a part of the picture which can be animated independently of other parts, for example a character's head or legs.

**aliasing**

Aliasing artifacts are introduced when a continuous signal is represented by discrete samples. Aliasing can manifest itself in jagged edges and Moire patterns in the spatial domain as well as strobing and speed distortion in the temporal domain.

**animation stand**

A conventional animation stand consists of a single frame animation camera mounted vertically above a table which can be moved north-south and east-west as well as rotated. Artwork is attached to a set of registration pegs mounted on bars which can slide in an east-west direction on the table. All movements are usually calibrated to an accuracy of 1/100".

**bi-packing**

Bi-packing is an optical printing process in which two pieces of film are loaded into a camera at the same time. One piece has previously been exposed and developed and thus acts as a mask, preventing the incoming light from striking certain regions of the second piece of film which is thereby selectively exposed.

**camera effect**

The most common camera effects in 2-D animation are zooms (changing the scale of the image) and pans (changing the position of the image). In 3-D animation more spectacular effects can be obtained by tracking (moving along with an object) and dollying (moving towards or away from an object).

**cel**

A cel is a piece of acetate (*celluloid*) on which animators paint parts of the image to be animated, for example a character's hands, arms, body, legs, or head. Up to half a dozen cels may be overlaid to produce the final composite image. By separating the animation into several cel layers the animator can save some time because only the parts of the image which actually move in a particular frame have to be redrawn.

**click**

When inbetween frames are generated by a linear interpolation system, the animation is smooth between keys, but at each key the direction and speed of motion change suddenly. This discontinuity is clearly noticeable as a visual click in the animation. The problem can be made less apparent by increasing the number of keys, but can only be solved completely by using an interpolation algorithm which provides continuity at the keys.

**colour lookup table**

Colour lookup tables are used to map the colour space used by the animator (for example, colour numbers between 0 and 63) into the colour space used by the system output hardware (for example red, green, and blue intensities in the range from 0 to 255). This translation mechanism makes colour selection much easier because the animator initially specifies only colour numbers without defining which intensities will eventually be used. After a test run on film any required colour changes can be made by simply updating the colour lookup tables.

**composition**

A composition is a collection of several independent sequences which run in parallel and/or consecutively. For example, a composition might consist of a background sequence, three sequences containing different characters, and one foreground sequence. The first two characters might appear at the same time while the third character only enters after the other two have left.

**critical flicker frequency (CFF)**

When the human eye is presented with a light source which is turned on and off extremely rapidly, the brain is not able to distinguish the light and dark phases and sees a continuous light. If the flicker frequency is lowered, at some point the light is no longer perceived as a continuous stimulus. For humans this critical flicker frequency lies around 45 Hertz, but its exact value depends on brightness, contrast, and other factors.

**dope sheet**

Dope sheets are used to specify the actions in an animated sequence. The sheet contains one column for each action (for example a body column, leg column, head column, and arm column) and one line for each frame. Camera effects and other special instructions are indicated with graphic codes such as arrows, lines etc.

**inbetween**

When developing a sequence the animator first draws only a small number of frames which correspond to important points in the action. These key frames are given to an assistant (*inbetweener*) who creates the intermediate drawings required to produce smooth animated motion. Computer assisted inbetweening systems use some mathematical formula (usually straightforward linear interpolation) to simulate the function of the human inbetweener.

**language driven system**

In a language driven system the animation is specified using some form of command language. The language may be a graphic extension to an existing programming language, or it may be a very simple set of commands which can be put together to form a script defining the desired action (see also *picture driven system*).

**level**

In an animation system which does not have full 3-D capabilities some illusion of depth can be provided by the use of levels. The animator assigns a level number to each cel in the sequence. This level determines the priorities for hidden surface processing. The technique can be useful in many applications but the fact that levels are static and must be defined by the animator makes them a nuisance whenever objects move in such a way that the level number must change (for example when a character turns around).

**line test**

Once the keys and inbetween frames have been drawn, the pencil sketches are filmed as a line test before any drawings are transferred to cels or coloured. The line test allows the animator to catch any errors in the animation before a lot of time is invested in the inking and opaquing phases.

**menu**

A menu is a list of options or commands which the animator can invoke by simply selecting one of the items from the menu, usually by pointing to it using a tablet and stylus.

**motion blur**

When a live action camera photographs a moving object, the individual frames are slightly blurred because the camera shutter stays open for typically 1/50 second. To avoid *strobing* in animation, motion blur is sometimes added artificially by drawing speed lines trailing fast moving objects.

**moving point constraints**

Moving point constraints [Reeves 81] allow the animator to control the way in which a keyframe animation system generates the intermediate frames. The animator can select important points in the picture and prescribe a motion path for each of these points. The inbetween frames are then calculated in such a way that all of the selected points follow their constrained paths while the unconstrained points are blended in, based on the neighbouring constrained points.

**multiplane camera**

The multiplane camera, invented by Walt Disney in 1936, allows the animator to place several layers of artwork on plates of glass which are mounted at different heights below an animation camera. Each level is lit and manipulated independently, and with careful planning a convincing illusion of three-dimensional space can be created with this technique.

**opaquing**

In traditional cel animation the animator's and inbetweeners' pencil sketches are first cleaned up and the outlines are then drawn with ink on sheets of acetate (cels). The cels are now turned over and the areas inside the outlines are filled with opaque colours from the back. This procedure is called opaquing, and the term has by analogy been transferred to computer animation systems where the cels are coloured by touching a point inside an area and asking the system to flood that area with a particular colour.

**outline**

Outlines are used to define areas of the image which are to be filled in with solid colours. In computer animation outlines can be filled by scan converting the area, calculating the intersections of the outline edges with all scanlines in the vertical range of the area. Alternatively, if the image is represented in a raster format, an area flood algorithm may be used. In this case the animator first selects a seed point inside the area, and the system then proceeds to fill adjacent points ("pixels") on the raster up to the enclosing boundary pixels.

**P-curve**

P-curves [Baecker 69a,b,c] allow the animator to specify a movement graphically by drawing a path along which he wants an object to move. By sampling the tablet at regular intervals while the path is being entered, the system can capture not only the shape of the path but also its dynamics (accelerations, hesitation etc.).

**picture driven system**

In a picture driven system the animator specifies the animation graphically rather than through the use of an animation language. Examples of picture driven systems include keyframe animation, P-curves, moving point constraints, and skeletons (see also *language driven system*).

**skeleton**

Skeletons allow the animator to control the motion of complex images by using simple stick figure representations. The stick figures are manipulated to generate the desired key positions, and when the system interpolates the inbetween frames, it automatically transforms the detailed pictures to match the animation defined by the skeleton.

**storyboard**

The storyboard in animation is a graphical equivalent of the script in live action cinematography. It usually consists of a large number of small sketches depicting all the scenes in the film.

**strobing**

In animation the term strobing is used to describe a phenomenon which occurs when an object is moved across the field of view too quickly. Instead of seeing the object move smoothly across the screen, the viewer will see it skipping from one position to the next. This effect is caused by *temporal aliasing* in the animation. The sample positions which were photographed to represent the motion are too far apart to allow the viewer to fuse them into a continuous motion. The problem can be alleviated by adding artificial *motion blur* to the animation.

**stroke**

When entering keyframes using the tablet and stylus, the animator controls the way in which the system will generate the intermediate frames by breaking each drawing into a number of strokes. A stroke is started when the stylus is depressed on the tablet and terminated when the stylus is lifted again. The system generates inbetweens by transforming corresponding strokes in the source and destination cels, i.e. by gradually changing the first stroke from the source cel into the first stroke of the destination cel etc.

**temporal aliasing**

Temporal aliasing occurs when a continuous phenomenon (such as smooth motion) is represented by discrete samples in the time dimension, for example a sequence of motion picture frames. The samples often cannot represent the phenomenon adequately, and the results may be backwards rotations of wheels, speed distortions, and other artifacts. These problems are usually more pronounced in animation because each frame represents only an instant in time whereas a live action frame captures the motion over a time period of typically 1/50 second.

## **APPENDIX B**

### **Bibliography**

- [Baecker 69a] R.M. Baecker. "Interactive Computer-Mediated Animation." Project MAC Technical Report MAC-TR-61 (Thesis), MIT, 1969.
- [Baecker 69b] R.M. Baecker. "Picture Driven Animation." *Proceedings AFIPS Spring Joint Computer Conference* 34, pp. 273-288, 1969.
- [Baecker et al. 69c] R.M. Baecker, L. Smith, and E. Martin. "GENESYS: An Interactive Computer-mediated Animation System," film available from Digital Computers Group, MIT Lincoln Lab., Lexington, Mass. 02173, 1969.
- [Booth et al. 82] K.S. Booth, D.H.U. Kochanek, and M. Wein, "Computer Animation in the 80's," *IEEE Spectrum* 1982. (submitted for publication)
- [Burtnyk & Wein 71] N. Burtnyk and M. Wein, "Computer Generated Key Frame Animation," *Journal of the SMPTE* 80, pp. 149-153, March 1971.
- [Burtnyk & Wein 76] N. Burtnyk and M. Wein, "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation," *Communications of the ACM* 19(10), pp. 564-569, October 1976.
- [Catmull 78] E. Catmull, "The Problems of Computer-Assisted Animation," *Computer Graphics (Siggraph '78)* 12(3), pp. 348-353, August 1978.
- [Clarke & Tyler 80] C.G. Clarke and T.W. Tyler, "American Cinematographer Manual," American Society of Cinematographers, Hollywood, 1980. (Fifth Edition)
- [Conte & DeBoor 80] S.D. Conte and C. DeBoor, "Elementary Numerical Analysis: An Algorithmic Approach," McGraw-Hill Book Company, New York, 1980. (Third Edition.)
- [DML 79] DML, "Animator's Manual for the Computer Generated Animation System," AM-DML-0778/79, Digital Methods Ltd., April 1979.
- [Evans et al. 81] K.B. Evans, P.P. Tanner, and M. Wein, "Tablet-Based Valuator that Provide One, Two, or Three Degrees of Freedom," *Computer Graphics (Siggraph '81)* 15(3), pp. 91-97, August 1981.
- [Foides 73] P. Foides, "Hunger (La Faim)," film available from the National Film Board of Canada, 1973.
- [Foley & Van Dam 82] J.D. Foley and A. Van Dam, "Fundamentals of Interactive Computer Graphics," Addison-Wesley Publications, Massachusetts, 1982.
- [Foley & Wallace 74] J.D. Foley and V.L. Wallace, "The Art of Natural Graphic Man-Machine Conversation," *Proceedings of the IEEE* April 1974.
- [Forsythe et al. 77] G.E. Forsythe, M.A. Malcolm, and C.B. Moier, "Computer Methods for Mathematical Computations," Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1977.
- [Hackathorn 77] R.J. Hackathorn, "ANIMA II: A 3-D Color Animation System," *Computer Graphics (Siggraph '77)* 11(2), pp. 54-64, Summer 1977.
- [Halas & Manvell 78] J. Halas and R. Manvell, "The Technique of Film Animation," Focal Press, London, 1978. (Ninth Edition)

- [Kitching & Emmett 76] A. Kitching and C. Emmett, "The Antics Computer System," *Computer Graphics* pp. 13-17, **10(4)**, Winter 1976.
- [Kovacs et al. 82] B. Kovacs, N.I. Badler, D.Zeltzer, L. Williams, and C. Reynolds, "Three-Dimensional Computer Animation," *Siggraph '82 Tutorial No. 12* ACM, July 1982.
- [Laybourne 79] K. Laybourne, "The Animation Book: A Complete Guide to Animated Filmmaking from Flip-books to Sound Cartoons," Crown Publishers Inc., New York, 1979.
- [Levoy 77] M.A. Levoy, "A Color Animation System Based on the Multiplane Technique," *Computer Graphics (Siggraph '77)* pp. 54-64, **11(2)**, August 1977.
- [Madsen 69] R. Madsen, "Animated Film: Concepts, Methods, Uses," Interland Publishing Inc., New York, 1969.
- [Max & Blunden 80] N. Max and J. Blunden, "Optical Printing in Computer Animation," *Computer Graphics (Siggraph '80)* pp. 171-177, **14(3)**, August 1980.
- [Parke 80] F.I. Parke, "Adaptation of Scan and Slit-Scan Techniques to Computer Animation," *Computer Graphics (Siggraph '80)* pp. 178-181, **14(3)**, August 1980.
- [Pearson 75] D.E. Pearson, "Transmission and Display of Pictorial Information," Pentech Press, London, 1975.
- [Ralston 65] A. Ralston, "A First Course in Numerical Analysis," McGraw-Hill, New York, 1965.
- [Reeves 81] W. Reeves, "Inbetweening for Computer Animation Utilizing Moving Point Constraints," *Computer Graphics (Siggraph '81)* **15(3)**, pp. 263-269, August 1981.
- [Smith 78] A.R. Smith, "Paint," Technical Memo No. 7, Computer Graphics Laboratory, NYIT, Old Westbury, New York, July 1978.
- [Smith 79] A.R. Smith, "Tint Fill," *Computer Graphics (Siggraph '79)* **13(2)**, pp. 276-283, August 1979.
- [Stern 79] G. Stern, "Softcel - An Application of Raster Scan Graphics to Conventional Cel Animation," *Computer Graphics (Siggraph '79)* **13(2)**, pp. 284-288, August 1979.
- [Thomas & Johnson 81] F. Thomas and O. Johnson, "Disney Animation: The Illusion of Life," Abbeville Press, New York, 1981.
- [Tuori 77] M.I. Tuori, "Tools and Techniques for Computer-aided Animation," M.Sc. Thesis, Department of Computer Science, University of Toronto, 1977.
- [Wallace 81] B.A. Wallace, "Merging and Transformation of Raster Images for Cartoon Animation," *Computer Graphics (Siggraph '81)* **15(3)**, pp. 253-262, August 1981.
- [Wendroff 69] B. Wendroff, "First Principles of Numerical Analysis: An Undergraduate Text," Addison-Wesley, Reading, Massachusetts, 1969.



## APPENDIX C

### Source Listings for Interpolation Procedures

```

procedure Find_poly (source, dest, next : integer);
{
* This procedure finds the weighting coefficients for the basis
* functions for the next interpolation interval. The key frames
* are stored in the array pix[max_keys][max_points]. The input
* arguments for this procedure are the indices for the source,
* destination, and next key.
* The coefficients found by this procedure are:
*   A = data value at source point
*   B = data value at destination point
*   C = derivative at source point
*   D = derivative at destination point
* Only the D-coefficient must be computed from scratch for each
* interval because new A = old B, new C = old D, new B = data
* value at destination. The D-coefficient is computed as the
* derivative of the parametric parabola interpolating the source
* point, destination point, and next point beyond the current
* interval.
* If no parabola can be found due to repeated points or lack of
* further data points for the first and last interval of a
* sequence, the derivative is set to 0.
}

var k: integer;
    dx_1, dy_1, dz_1, dx_2, dy_2, dz_2 : real;
    delta_1, delta_2 : real;
    del_q1, del_q2 : real;

begin
  if cur_key = 2 then {set up for the first interval}
    for k := 1 to n_points[dest] do
      begin
        x_poly[k].B := pix[source][k].x;
        y_poly[k].B := pix[source][k].y;
        z_poly[k].B := pix[source][k].z;
        x_poly[k].D := 0.0;
        y_poly[k].D := 0.0;
        z_poly[k].D := 0.0;
      end;
  end;

```

```

for k := 1 to n_points[dest] do
begin
    { Set up A, B, and C coefficients from previous data }

    x_poly[k].A := x_poly[k].B;
    y_poly[k].A := y_poly[k].B;
    z_poly[k].A := z_poly[k].C;
    x_poly[k].B := pix[dest][k].x;
    y_poly[k].B := pix[dest][k].y;
    z_poly[k].B := pix[dest][k].z;
    x_poly[k].C := x_poly[k].D;
    y_poly[k].C := y_poly[k].D;
    z_poly[k].C := z_poly[k].D;

    { Find D-coefficients by calculating the derivatives of
      the interpolating parabolas for each coordinate }

    dx_1 := pix[dest][k].x - pix[source][k].x;
    dy_1 := pix[dest][k].y - pix[source][k].y;
    dz_1 := pix[dest][k].z - pix[source][k].z;
    dx_2 := pix[next][k].x - pix[dest][k].x;
    dy_2 := pix[next][k].y - pix[dest][k].y;
    dz_2 := pix[next][k].z - pix[dest][k].z;

    if (cur_key <> n_keys)
      and ((dx_1 <> 0.0) or (dy_1 <> 0.0) or (dz_1 <> 0.0))
      and ((dx_2 <> 0.0) or (dy_2 <> 0.0) or (dz_2 <> 0.0)) then
    begin
        delta_1 := sqrt (dx_1*dx_1 + dy_1*dy_1 + dz_1*dz_1);
        delta_2 := sqrt (dx_2*dx_2 + dy_2*dy_2 + dz_2*dz_2);
        del_q1 := delta_2 / delta_1;
        del_q2 := delta_2 / (delta_1 + delta_2);
        x_poly[k].D := dx_2 + ((dx_1*del_q1) - dx_2) * del_q2;
        y_poly[k].D := dy_2 + ((dy_1*del_q1) - dy_2) * del_q2;
        z_poly[k].D := dz_2 + ((dz_1*del_q1) - dz_2) * del_q2;
    end
    else begin

        { Special case: repeated point. Set derivatives to 0.0 }

        x_poly[k].D := 0.0;
        y_poly[k].D := 0.0;
        z_poly[k].D := 0.0;
    end;
end;
end; { procedure Find_poly }

```

```

procedure Eval_poly (k:integer; s:real; var x, y, z:integer);
{
* Evaluate the interpolated x, y, and z value for the k'th point
* in the picture at time s. First find the values of the four
* contributing basis functions at time s, then find the total
* polynomial value by multiplying each basis function by its
* associated coefficient.
}

var basis : array [1..4] of real;
    s_sqr, s_cube : real;

begin
    s_sqr := s * s;
    s_cube := s * s_sqr;
    basis[1] := 2.0 * s_cube - 3.0 * s_sqr + 1.0;
    basis[2] := -2.0 * s_cube + 3.0 * s_sqr;
    basis[3] := s_cube - 2.0 * s_sqr + s;
    basis[4] := s_cube - s_sqr;

    x := round (basis[1] * x_poly[k].A
                + basis[2] * x_poly[k].B
                + basis[3] * x_poly[k].C
                + basis[4] * x_poly[k].D;
    y := round (basis[1] * y_poly[k].A
                + basis[2] * y_poly[k].B
                + basis[3] * y_poly[k].C
                + basis[4] * y_poly[k].D;
    z := round (basis[1] * z_poly[k].A
                + basis[2] * z_poly[k].B
                + basis[3] * z_poly[k].C
                + basis[4] * z_poly[k].D;

end; { procedure Eval_poly }

```

**APPENDIX D**  
Demonstration Film

Title: "Interpolation Courbe - Demonstration I"  
 Format: 16mm film, black & white, silent  
 Length: 5 min.  
 Produced by: National Film Board of Canada  
 Date: April 1982

This demonstration film contains six different sequences showing the characteristics of the curved interpolation approach as compared to straightforward linear interpolation. All sequences follow the same format. First the key positions for the sequence are shown, superimposed in a single image. Next, the animation produced by linear interpolation is shown on the left side of the screen, followed by the curved interpolation on the right side of the screen. Each sequence concludes with a side by side comparison of the two methods.

In all cases the curved interpolation is based on exactly the same keys as the linear interpolation with the same number of inbetweens. No additional information was given. This section provides some technical details for each sequence and discusses some of the effects which can be observed.

### List of Sequences

- | Title  |     |
|--|-----|
| 1. Automatic Acceleration and Deceleration     | 8 1 |
| 2. Zig-Zag / Sinusoidal Motion                 | 20  |
| 3. Octagonal / Circular Motion in 2 Dimensions | cy  |
| 4. Octagonal / Circular Motion in 3 Dimensions |     |
| 5. Metamorphosis of Squares in 3 Dimensions    |     |
| 6. Floating Text in 3 Dimensions               | oc  |
| Credits  | pa  |

#### Sequence 1: Automatic Acceleration and Deceleration

2 key positions  
 30 inbetween frames in each interval  
 cycle repeated for 10 seconds

In this sequence an object moves back and forth between two key positions. In this case both interpolation programs produce inbetweens along a straight line. However, the motion dynamics differ significantly. In the linear interpolation the object moves towards the next key at a constant speed, e.g. speed = +s. At the key it is suddenly forced to move in the opposite direction (speed = -s). No acceleration or deceleration occurs to absorb the shock of the speed reversal.

In the curved interpolation the continuity constraints at the keys imply that the speed cannot suddenly change from +s to -s, and therefore the object decelerates to a stop first (speed = 0) before accelerating in the opposite direction. These accelerations and decelerations are supplied automatically by the interpolation algorithm, and no animator intervention is necessary.

### **Sequence 2: Zig-Zag / Sinusoidal Motion**

5 key positions  
20 inbetween frames in each interval  
cycle repeated for 15 seconds

In this sequence the keys are positioned at the corners of an "M". The linear interpolation produces a zig-zag path where the direction of motion changes abruptly at each key. This sequence clearly demonstrates the problem of "clicks" in linear keyframe interpolation. The curved path interpolation moves the object along an almost sinusoidal path with no discontinuities in the direction of motion.

### **Sequence 3: Octagonal / Circular Motion in 2 Dimensions**

8 key positions  
20 inbetweens in each interval  
cycle repeated for 13 seconds

The eight keys in this sequence are placed on the corners of a 2-dimensional octagon. In the linear interpolation the object moves exactly along an octagonal path while in the curved interpolation the movement is smoothed to give a close approximation to a circular path without having explicitly specified that the desired motion is a form of rotation.

### **Sequence 4: Octagonal / Circular Motion in 3 Dimensions**

8 key positions  
15 inbetweens in each interval  
cycle repeated for 13 seconds

This sequence shows an effect similar to Sequence 3, i.e. octagonal as compared to almost circular motion. Because the orientation of the object changes as it moves, the distortion phenomenon illustrated in Figure 2.22 is very visible. The object shrinks significantly between the keys in the linear interpolation, but in the curved interpolation this problem does not occur.

**Sequence 5: Metamorphosis of Squares in 3 Dimensions**

9 key positions

30 inbetweens in each interval

This sequence demonstrates the animation of an object which is in constant metamorphosis and at the same time moves in 3-D space. The linear interpolation is discontinuous ("clicks") in terms of the quality of both the metamorphosis and the 3-D path. The curved interpolation is sufficiently smooth to make it nearly impossible to differentiate between the key positions and the inbetween frames generated by the system.

**Sequence 6: Floating Text in 3 Dimensions**

9 key positions

30 inbetweens in each interval

cycle repeated for 23 seconds

This sequence shows the movement of a rigid object along a fairly complex path in 3 dimensions. In the linear interpolation the keyframes are clearly visible while the curved interpolation produces a uniformly smooth motion. Some additional tests showed that to obtain a similar quality of motion using linear interpolation the number of keys had to be increased from 9 to at least 40...

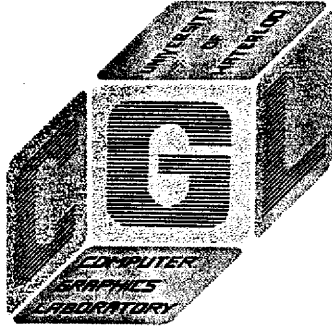


The following technical reports are in preparation by members of CGL and will be available by December 31, 1982.

- CS-82-41 A Graphics Editor for Benesh Dance Notation  
*Singh, Beatty, Booth & Ryman*
- CS-82-42 A Computer System for Smooth Keyframe Animation  
*Kochanek, Bartels & Booth*
- CS-82-43 Frame Buffer Animation  
*MacKay & Booth*
- CS-82-44 Colour Principles and Experience for Computer Graphics  
*Goetz & Beatty*
- CS-82-45 A Powerful Interface to a High-Performance Raster Graphics System  
*Breslin & Beatty*
- CS-82-46 Picture Creation Systems  
*Plebon & Booth*
- CS-82-47 Anthropomorphic Programming  
*Booth, Gentleman & Schaeffer*
- CS-82-48 A Scene Description Language  
*Lea & Booth*
- CS-82-49 Varying the Betas in Beta-splines  
*Barsky & Beatty*

Reports in stock are forwarded free of charge. A nominal fee is charged for out of stock items. For an up-to-date listing of available reports, please write to

Technical Reports  
Department of Computer Science  
University of Waterloo  
Waterloo, Ontario, Canada N2L 3G1



*The Computer Graphics Laboratory at the University of Waterloo is a group of students and faculty doing research within the Computer Science Department. Interests include most areas of computer graphics, document preparation and man-machine interfaces. CGL is affiliated with the Institute for Computer Research. Graduate work at the masters and doctoral level leads to a Computer Science degree from the Faculty of Mathematics. Further information may be obtained by writing to the following address:*

*Computer Graphics Laboratory  
University of Waterloo  
Waterloo, Ontario, Canada N2L 3G1  
(519) 886-1351*