Recommendations Towards a Direct Manipulation Interface for Three-Dimensional Computer Animation

by

CRAIG ARTHUR ORCUTT

B.S., Kansas State University, 1987

A THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Computer and Information Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

Approved by:

W H Hankley

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

I thank Dr. William J. Hankley for his careful guidance; Drs. John J. Devore and Virgil Wallentine for their helpful suggestions; and my wife, Cheryl, without whose patience and criticism I would not have been able to finish this work.

1. Introduction

1.1. Overview and Purpose

The art of animation has seen many technological breakthroughs in the past. From the advent of the first filming of cartoons through the development of new techniques and practices, the art has grown in popularity and complexity. Not the least important advance is the usage of computers to aid in the animation process. There have been several animation systems [3,4,6-9,16,19] written for computers, with varying degrees of success. These systems were either aids used in the traditional activities of the animator, or were the medium for the animation itself. The latter class of systems appears more interesting because they allow the animator to perform tasks in animation that were not previously or physically possible. One drawback of these systems, however, is that the animator must be more aware of the computer than is necessary. This problem is usually manifested in the form of a computer director's language that the animator must either partially or fully use to perform the desired motion. This is neither an intuitive nor cognitively bound technique to the activity to which animators are familiar. Animators typically manipulate models, not program computers.

The direct manipulation interface is a recent design technology, and valuable interface tool [20], through which

actions performed on the computer are directly analogous to actions performed, physically, in the real world. Natural activities are transferred to the interface so that people can use the computer in a fashion that is immediately recognizable and intuitive to them. This makes the learning process much quicker and more efficient. Also, people tend to enjoy working with this type of interface more than with a simple command driven interface.

The major contribution of this work is to present a functional union of the direct manipulation interface ideology with three-dimensional computer animation through a detailed set of recommendations that may be used towards an eventual design. No effort was made to provide a structured design, nor to suggest usable data structures.

The paper is organized in the order of increasing detail. The remainder of the introduction details the traditional practices of both two and three-dimensional animation and the advantages and hindrances of computer systems for animation. Chapter two details how several different animation-oriented computer systems have implemented these practices. Chapter three covers goals that were considered and are followed throughout this paper. The presentation of the recommendations begins with chapter four, where the basic look and feel is discussed. Chapter five annotates the procedures for the static manipulation of the animatable models. Lastly, chapter six proceeds with the dynamic manipulation of those models. Chapter seven presents major

conclusions and directions for future research.

1.2. Traditional Animation and the Computer

Experts [1,2] agree that computer assisted animation is well divided into two main areas, two-dimensional and three-dimensional. The computer itself assists in functions of these two divisions in a wide array of positions from management tools to fully integrated animation systems; from computer-assisted animation to computer-controlled animation. This section covers the traditional practices of two-dimensional animation, and then proceeds on to the three-dimensional variety.

Classical two-dimensional animation is performed using a time-proven series of steps. First, a **storyboard** is created depicting the major scenes. The storyboard consists of a series of **key frames:** raw, unfinished drawings which actually are to be incorporated into the finished product. Once the general flow of animation has been decided upon, several levels of assistant animators draw the frames that exist in between each of the key frames. This process, called **in-betweening**, is the most tedious and costly, thus the most likely candidate for computer assistance. After all of the individual frames have been drawn, they are either manually or electronically transferred to clear sheets of acetate called **cels.** The cels are then **opaqued**, or colored in, another time consuming process. After all of the cels are

ready, they are mounted in an **animation stand** and photographed. Cels can be placed on top of others to form complex images.

Computers can and have been used in this process for each of the above steps, and in each step either as an assistant, or as the medium of animation. Most particularly, the in-betweening and opaquing phases have seen the most computer integration in the past. Some systems, as well, have been designed for both the transfer and opaquing steps [3], and for the photographing step [4].

The major problems with either computer-assisted or computer-controlled, two-dimensional animation are basically caused by the fact that this type of animation only has two dimensions [5]. For example, in the in-betweening phase it is necessary for the computer to be able to interpolate one or more intermediary frames in between two given key frames. The interpolated frames would represent the animated characters in various stages of motion. For most types of motion such as rotations, however, not enough information is located in the available key frames for effective interpolation, ie. arms that are covered by bodies in key frames could be extended through in-between frames, yet the computer does not have enough information about the arm to compute interpolations, because it only has the key frames to work with. Several solutions to this problem have been proposed [5], yet none work in all cases, and most require the animator to define more of the picture; to tell the

computer the information which it needs to know. This extra requirement of the animator is neither welcomed nor extremely useful. This continues to be an research topic at this time.

For the large part, many of these problems simply do not exist for animators working in a three-dimensional medium [1]. In classical three-dimensional animation, real models are created for each character in motion. For each successive frame of animation, the animator painstakingly adjusts each model a slight amount, so that when the film is played back, the models appear to be moving.

This type of animation is very amenable for integration and control by computers. Since the models are three-dimensional and not two, the computer has all of the information it needs to perform the in-betweening steps of animation. Models, also, are well representable by computers, and even complex three-dimensional wire-frame animation can be achieved in real time by powerful processors. The animator is relieved of the burden of hand-adjusting models, and is simply asked to inform the computer as to what motion the models are to undergo. The method by which the animator is to inform the computer about animation is the main focus of this paper.

2. Survey of Current Animation Systems

This chapter covers five of the most notable computer animation systems actively available and currently in use. They are arranged by order of appearance. As a brief overview, programming languages are currently the most prevalent medium for the control of animation by a computer. These languages are most often concerned with three-dimensional animation because it is inherently more controllable by a computer.

2.1. GRAMPS

In 1981, a computer animation language was introduced by a national chemistry research organization to be able to display even the most complex chemical reactions and transformations on a video screen in real time. This language, called GRAMPS [8] for GRAphics for the Multi-Picture System, utilizes an assembler-like language and specialized hardware to let a user adjust the display image. This configuration actually makes GRAMPS more versatile and capable of handling more than just chemistry-oriented animation. By manipulating dials, a joystick and a graphics tablet, a user can interactively view any stage of a chemical process desired. Being one of the first attempts at writing an animation language, GRAMPS is oriented specifically towards programmers. Professional animators cannot use the system unless

they are, themselves, programmers. Most of this problem arises from the fact that the language is strictly procedural and not pictorial. Also control statements are missing, so that only simple animation sequences can be viewed and manipulated. Also, no image rendering is provided for the viewer. GRAMPS does, however, allow real-time interaction with the animation.

2.2. ASAS

Reynolds' language, ASAS [7] (Actor/Scriptor Animation System,) is one of the first animation languages to be used for professional production of movies. The movie TRON was created using animation effects generated using this language. ASAS is actually an extension to the LISP programming language, introducing various data types and control statements to facilitate the creation of three-dimensional animation. This language is, again, procedural, yet this time parallelism is introduced by allowing the declaration of actors, discrete animated units, before they are actually animated. Several actors could then be animated during similar time spans; time being represented strictly using frame numbers. ASAS also introduces message passing for explicit synchronization. Like GRAMPS, use of this system by professional animators would be difficult since the language is strictly procedural and not pictorial. The author states that this is a necessity for professional looking

computer controlled animation, but the statement does not seem necessarily true.

2.3. DIAL

One of the first efforts to create an animation language that is more pictorial is DIAL [6], for DIagrammatic Animation Language. In this language, each animated object is still declared procedurally, but are executed in parallel using a pictorial syntax. This syntax is a slight extension to the GANTT chart methodology, and is actually very analogous to music scores. Starting and stopping times are explicitly shown and intuitively more obvious to the programmer. Even though the professional animator must still know how to program, the requirement is not as stringent with DIAL, and the animator does have some analogous indication to what he is used to in his art: parallelism of animated objects.

2.4. TWIXT

TWIXT [19] is an interactive, computer graphics animation system that is currently in use at the Computer Graphics Research Group of the Ohio State University. The system was created in response to the fact that prior animation systems require the use of a high-level programming language to implement motion. TWIXT allows the animator to visually and interactively modify the animation sequences at any

point in time. This permits the animator to see images as they are dynamically modified.

TWIXT is not a key-frame system, one that rapidly plays 30 still frames per second to produce the animation. This type of animation is not amenable to modification. Rather, a multiple track system was used. Each object in motion, and each attribute of these objects, can each be given its own unique track. The tracks represent dynamic changes in display parameters, such as position, orientation, color, etc. When all of the active tracks are played at the same time, animation occurs. This also means that more computation is required to play back the animation. The animator, then, is given the choice of real-time, wire-frame playback or full image-rendered playback via some recording device.

The time unit used in TWIXT is the frame number. Seconds is not used as a time unit because it is not traditionally used in the field. Tracks are created using frame number references. In-between frames are then interpolated using a variety of interpolation schemes.

The user interface is interactive, inasmuch as what the animator is currently controlling is dynamically visible on the computer screen. However, the animator must still learn a large list of command words to manipulate the viewed objects. This is only a one-step improvement over the usage of a high-level animation language.

2.5. MIRANIM

In 1985, a great advance was made towards creation of a truly user-oriented computer system for the control of animation. This system, Miranim [9], was created specifically for the animator, not the programmer. Through the interactive use of menus, the animator can create objects using geometric and parametric shapes. Several shapes can be combined together to create a background, or to create an actor, which is an entity of motion. The animator can then define one or more cameras, which are capable of special effects, and one or more light sources, each with their own diffusion and spectrum characteristics. After the above entities are defined, each is given motion directions, start and finish times and position requirements and then the animation can be viewed. If the desired motion parameters are too complex to describe in the limited fashion offered the animator, he can give the motion specifications to a programmer who will be able to use a computer animation sublanguage to mathematically, and non-intuitively, describe the action. The interactive user-oriented aspects of Miranim are good and admirable, but still require the command driven interface. The animator is still required to learn a director syntax, and know the complex mathematical equations which define motion.

In one example [17] of creating animation using a procedural language offered by Miranim, Magnenat-Thalmann men-

tions that creating animation using an animation language is extremely time consuming. She states that a 13 minute animated film, Dream Flight, took 14 months to produce. However, they also point out that user-friendly interactive systems decrease the amount of time to create an animated sequence, but also limit the creativity of the animator. This is probably a limitation of the interactive system, however, not of the animator. A well designed interface should not have this problem.

3. Interface Design Goals

The ideal interface for a user-oriented computer controlled animation system must be designed and developed using the following goals. One expert [1] notes that the best user-interface for animation is the one that the animator is already used to. A system should be developed around the artist, not the programmer.

3.1. Expert Opinions

Some general goals in the creation of an effective user interface are covered by Foley et al [15]. Interactive graphics should be benign, responsive and graphic; it should behave well. Also, a good interface removes the computer system from conscious concern by the user, letting the artist concentrate on art. Some of the measures of the quality of a user interface design are the time a user must spend accomplishing a task, the accuracy with which that task is completed and the pleasure the user derives from using the system.

Some specific goals for the design of an interactive animation system are presented by Csuri, et al in [16]. They recommend, first, that the interaction used to create the animation should ideally be in real-time, even though current hardware capabilities often fall short of satisfying performance requirements. Second, they state that a lan-

guage is required to control the transformation and interactions of the objects being animated. The main emphasis of this paper is to present a list of recommendations towards a future design using a direct-manipulation interface which makes this requirement less emphatic if not totally unnecessary.

As a third design consideration, the authors suggest using procedural models to describe complex objects. (PHIGS is an example of such a model.) They mention that, when interacting with the creation of the animation, usually a lower resolution image of the object(s) should be used to facilitate their real-time manipulation. Fourth, the system should have editing capabilities to adjust the animation. Fifth, the display algorithm should be able to handle the intersections of objects in three-dimensions to provide depth cues to the animator. Lastly, they mention that the display algorithm should be tightly associated with the hardware to increase performance and reduce complexity.

Swezey and Davis [18] offer the following human factors guidelines for computer graphics, in general. They note, interestingly, that there are no current guidelines specifically for computer animation.

The authors mention that the time that is required by the user to find information on the screen is directly related to the number of objects that are being displayed. They recommend that only the most relevant information should be on the screen at any time. Other information

should be removed once it is no longer needed. Then, users should be able to recall objects onto the screen. When the screen does become cluttered, or when symbols are similarly shaped, the system should offer zoom functions to concentrate on a single object or group of objects. This feature also separates the contiguous symbols which can later be viewed using a scrolling function. A selectable button (or any selectable area on the computer screen) should be at least 0.25 inch square.

3.2. Specific Design Goals

The following are specific, personal goals towards the creation of an effective user's interface for three-dimensional computer animation:

3.2.1. Pictorial Display

The creation of objects to move around, the relationships between objects, the definition of motion of these objects, etc., should all be done graphically as if the animator were actually performing these tasks. As many actions as possible that can by represented pictorially should be. No syntax, no matter how intelligent should be used.

3.2.2. Intuitive Display

Similar to the point above, the system should take real actions of the director and transfer them to the animation

system. If the director motions the actor to move a certain way, that same control should be offered in the animation system. If the director wishes a particular camera angle or special effect, or a particular lighting scheme, it should be made available. The view of the computer animation should also represent the view of reality as much as possible.

3.2.3. Production Time Improvement

One of the main reasons that computers are used in any production process is to decrease the amount of time required to perform a specific task. The ideal system should make the production of animation as quick as possible without compromising its quality.

3.2.4. Quality Improvement

With the integration of the computer, so comes the integration with all the power of the computer. Special effects and controls should be allowed the animator that are not normally available. The animation system should be as usable and as powerful as possible.

3.2.5. Extensibility

The system should be expandable, with no limits on any possibilities of animation the animator may desire. This probably would mean the integration of a computer sub-language as part of the system. If this is necessary, then the

sub-language should be as powerful as possible, but still intuitive and learnable by the animator. The definition of such a language has been covered elsewhere [6-9,14], and is beyond the scope of this paper.

4. The Interface

The purpose of this paper is to present a list of recommendations towards a design for an interface that will allow animators to naturally create their art on the computer. This is primarily possible through the concept of the direct manipulation interface. Since the most effective interface system currently in use that enhances this concept is the windowing system, it will be the basis for these recommendations. No effort was made to provide compatibility with nor to conform to any existing windows interface implementation. This was done to provide a more general format for the discussion of the proposed windows.

The windowing system was chosen for several reasons. For instance, any number of windows can be viewed at a time. This is a necessary requirement, since several perspectives of an animated scene will be possible. Also, much information will be provided through these windows, and at times it may become desirable to limit the amount of perceived information to ease the burden on the animator. This will be possible by simply removing any given windows from view.

Another reason is that the individual windows have dynamic characteristics. They can be adjusted in size, and even moved around on the computer screen, so that the animators will be able to organize the available information in a

manner that will be most convenient to them.

The windowing system is not, however, perfectly ideal. The system shares a basic quality with all computer displays. The quality is that the computer display is necessarily static. At any given point in time, the current state of the animation in progress is still. The problem becomes apparent when the animator will wish to edit an animation sequence, but will only be able to manipulate a single instance in time, not a moving scene. This may gives rise to some confusion, and the problem must be considered during the entire design process.

Every window in the interface will have certain common characteristics. This will mainly allow for the transference of knowledge from one aspect of the interface to all others, and is a basic methodology behind all windowing interfaces, in general. Each window will include the standard buttons, for expanding, closing, and positioning the windows. Also included in each window will be the scroll bars to adjust the current viewed scope.

The images viewed in each of the windows should ideally be in full image-rendered form, with color, texture and patterns included. This, however, is beyond the power and capability of current hardware, so simple line-drawings with hidden lines and surfaces removed should be adequate.

The individual differences in the appearances and pur-

poses of the windows is discussed below. Following that will be a presentation of the pull-down menus, offered as a supplement to the power of the windows in the interface. Lastly, an overview of the perceived hardware requirements will be given.

4.1. Scene Window

Each window described in this and following sections will be useful to the animator, but the scene window (Fig. 4.1) is easily the most beneficial. This window will be used to view the area upon which the animation is performed. It is also here that the gross manipulation of the animatable models will be accomplished. In the scene window, the animator will be limited to describing simple positioning motions with the models. He will be able to move the models through space, but not be able to adjust the models, themselves. This is necessary, since with all but the most simple animation sequences, there are many models, each of which can be finely adjusted. There will simply be too much information visible in the scene window to be able to quickly and easily perform those fine adjustments. This task, rather, will be performed in the actor window, which will be discussed shortly.

Every scene will be given a name, which is also the file name used when saving the scene. The label field at

the top of the scene window will contain the word ''Scene:'' and the name of the scene. More than one scene should be incorporatable into an animation sequence.

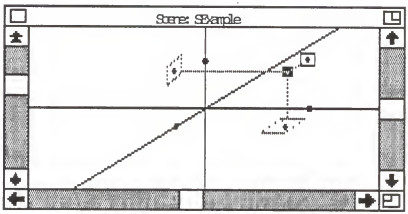

Fig. 4.1: The Scene Window

Through the scene window, the entire scene could be viewed, even those areas which are not the focal interests of the cameras. This will allow the animator to both set up the animation easily and view the progress of the entire scene. The potential degree of information overload, again, will be large in this window, so the animator will also be given the power to view part of the scene, but at a larger magnification. This is accomplished through the facility of a third scroll bar which will most conveniently be placed along the left side of the window. This scroll bar will let the animator adjust the viewing distance from the actual models, and otherwise has the full functionality of the two

normal scroll bars. The closer the viewing distance, the less that can be viewed, but also, the larger the models appear. The arrow buttons for this scroll bar must point first into and then out of the plane of the computer screen. If the resolution of the monitor is not great enough to show these clearly, then the words ''in'' and ''out'' may be substituted for the arrow symbols. If a windows interface implementation is used that cannot accommodate the third dimension scroll bar, then pull-down menu options could be used instead.

The selection of objects in the scene window could be easily performed with a two-dimensional selection device, in all cases except the one where the object of interest will be behind another and not viewable, and thus it would be impossible to point to it. The third dimension scroll bar could help in this matter by allowing the animator to zoom past the closer object. This will allow the object of interest to be viewable, and thus selectable.

Since the typical scene with which an animator works is three-dimensional, then so must be the scene which is perceived through the scene window. The window, however, is only two-dimensional, so a third dimension cuing model must be incorporated. The model incorporated for these recommendations is similar to that described by Thatch and Myklebust in [10]. The model uses a combination of four cues to allow

the animator to easily perceive his work in three dimensions. The first of these cues will be the coordinate axes that are drawn on the scene, but not viewable through the cameras. The axes will consist of orthogonal X and Y axes, and an origin intersecting, oblique Z axis. This is the most common view of a three-dimensional coordinate system, and thus seems the most appropriate.

The axes will be used to facilitate the perspective viewing enhanced by the next two cues. Both of these cues will show the position of an object of interest, which will most often be a selected model to be manipulated. The first of these is the axial cue, which will highlight a point on each of the three axes corresponding to the selected objects position. The other cue is planar, and will show a point on each of the XY, XZ and YZ planes corresponding to the position of the selected object. Each point will be connected to the selected object with a dotted line. Also, a dotted-line box, parallel to each of the axes, will be drawn around each planar point for emphasis.

The fourth cue will be provided in the form of the hidden line and surface removal performed in real-time in all visible windows. Objects that will be behind other objects appear to be behind them. This will perhaps be the most effective form of depth cue.

One major inconvenience to this cuing model will be

that the coordinate system is drawn along with the scene. The confusion that this may introduce can be eliminated simply by offering a pull-down menu option to hide the axes. The other two cues are only visible while an object is selected, so they can be turned off simply by deselecting all objects. An accompanying pull-down menu option to show the axes, should also be incorporated, since the other two cues are less effective without them.

4.2. Actor Window

The actor window (Fig. 4.2) can be simplistically thought of as a magnification of one of the models in the scene window. It will have this function, but it will also allow the animator to adjust the model of interest in a wide array of possibilities and with fine precision.

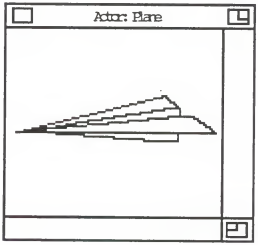

Fig. 4.2: The Actor Window

If possible, the model will be viewed fully in the actor window without necessitating clipping. This will eliminate the need for scroll bars. If the model is too large or too complex, however, for the easy selection and manipulation of even the most diminutive parts, then as much of the model will be displayed as possible, and the scroll bars will be included. The capability of dynamically resizing the actor window will, of course, be of great value in this respect.

Every model, or actor, will be given a name. While a model is viewed in the actor window, the label field will contain the word ''Actor:'' and the name of the actor.

While an object is selected in the scene window, it will be viewed in the actor window. It is through the actor window that most of the fine animation will take place. Note, however, that all changes performed to an model while it is in the actor window, will also be dynamically performed to the model's image in the scene window. This concept of dynamic updating will be incorporated throughout the interface. For instance, even during pencil tests of the animation, if an object is selected and viewed in the actor window, not only will the camera view be dynamically updated, but also the scene and the model in the actor window. This should greatly enhance the detailed perception of even the most complex animation sequences.

4.3. View Window

Even though the scene will be readily viewable using the scene window, animators will wish to select special perspectives and viewpoints using different cameras. In the actual art of animation, the animator may have any number of cameras filming different areas of the scene, or with different parameters. This interface is designed to allow any number of different cameras to film a scene.

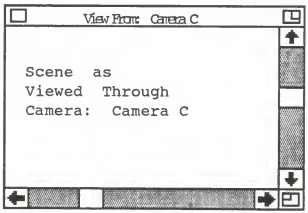

Fig. 4.3: A View Window

As will be discussed later, any number of logical cameras can be dynamically created. Each of these cameras could be used to view a scene differently, so each must have its own view window. A view window (Fig. 4.3) will be a dynamically updated representation of that portion of the scene at which its associated camera will be aimed. Again, this representation will probably only be a line drawing

approximation, because of hardware limitations. Any number of the view windows can be visible on the screen at a time.

Every camera will be given a name. While a view window is visible on the screen, the label field will contain the word ''Camera:'' and the name of the camera.

4.3.1. Global View Window

There will be one view window that will not be dedicated to just a single associated camera. This window will be called the Global View Window, and its label field will show ''Global View:'' which will be followed by the names of all of the current cameras. This window will give a line-drawing representation of the scene as it will appear on final film. This is different than the normal view windows, in that they will always show the scene as viewed through their associated cameras. At different times, different cameras may be selected to be the ones currently filming the scene. The global view window represents this final image. It also facilitates the viewing of special camera effects which are not viewable elsewhere in the interface.

Every view window, including the global view window, shows the image as it will be viewed at a given point in time. This time reference is part of the track window and will be discussed below. If the current time, or frame number, is adjusted in the track window, the images in all

view windows will be dynamically updated to reflect the new current states of the models in the scene.

4.4. Track Window

Many of the most recent and most powerful animation systems have utilized the multiple track concept of animation [6,19]. Through this concept, any number of model attributes can be assigned unique tracks and thus animated separately. This allows the animator more freedom with the art, and makes the editing of animation sequences simpler. As an added benefit, the multiple track concept facilitates

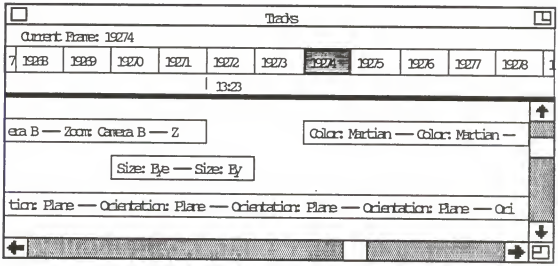

Fig. 4.4: The Track Window

the parallel motion of objects.

The multiple track concept will be incorporated into this interface through the use of the track window (Fig.

4.4.) The window consists of all the common attributes, and some added functionality. The top portion of the window will contain the frame indicators. This will be directly analogous to the film of an actual recording. Each frame will succeedingly numbered to act as references during the editing of the animation. This portion of the window should only be horizontally scrollable, and not vertically. The track indicators below the frame indicators should be vertically scrollable when desired.

Frame number references are used as the medium of time rather than minutes and seconds. This follows a long standing custom in the animation industry, and other systems [7,19] have shown that animators prefer it. However, minute and second designations will also be given. The typical speed of film is either 24 or 30 frames per second [19], but could be adjusted easily through the use of a pull-down menu.

A very important notion will be that of the current time. Every view window will display a representation of the scene at the current time. At all times, one of the frames must be designated as being the current one. It could be shown by darkening the area indicated on the frame indicators. Since the current view of the track window may be several frames away from the current frame, it could become difficult to find. For this reason, the current

frame number will also be displayed directly above the frame indicators.

The tracks, themselves, will be displayed below the frame indicators. Any number of tracks could be created and edited dynamically. Also, the animator should be able to save individual or groupings of tracks to a secondary storage device to be used later as a type of macro substitution process to make the entire routine of repetition in animation more manageable. The main purpose of the scroll bars will be to allow the scanning of the possibly large number of tracks in the window. The left side of a given track corresponds to the time frame when that particular animation will start. The animation will continue until the frame indicated by the right-most end of the track. In each track will be text that will give a description of the type of animation that is designated by that track. During the span of time that a particular animation is to occur, the computer will mathematically interpolate in-between positions for the object of interest. This is the one major factor that will save animators time. Several interpolation schemes should be offered, such as linear, ease-in, ease-out and ease-in-ease-out. These could be selectable from a pull-down menu. An animation that starts and stops on the same frame will be instantaneous, and no interpolation will be required. The track, itself, will contain all of the

pertinent information to perform the interpolation schemes.

The individual tracks will make the process of editing simple. The left and right ends of the tracks will be selectable, and the track could be stretched either direction in time. It could be deleted, as well.

4.5. Pull-Down Menus

Pull-down menus were incorporated into the standard windows interface to add functionality to the power of the windows themselves. These menus allow the user to perform filing, housekeeping or other actions. Several pull-down menus will be incorporated into this interface. Included in the list of menus will be those offered below and collections of other options designated throughout this paper.

The first pull-down menu will be for filing functions. It is here that animation sequences, scenes, tracks and actors could be saved to and loaded from a secondary storage device. When an actor is saved, no orientation or position information will be saved with it. On the other hand, when a scene is saved, all actors and objects in the scene, as well as all of their attributes will be saved. When a track is saved, only the time span and the action will be recorded, but when an entire animation sequence is saved, not only will all the information from the track window be saved, but also all of the information pertaining to the

actors and objects, as well.

In the filing menu, the user will also be able to open any of the scene, actor, view or track windows that were previously closed.

Also available in the filing menu are the options for actually animating the sequences. There will be two modes of animation, the pencil-test and fully image-rendered modes. These are the typical options offered on other systems [9,16,19]. The advantages of the pencil-test mode are that, in this mode, the animation could be viewed in real-time on the computer screen, itself. This will allow the animation to be quickly and easily seen, so that editing actions could follow. The disadvantages are that only line drawing approximations to the models will be seen, albeit with hidden line and surface removal. The fully image-rendered mode will incorporate all of the shading, lighting and coloring functions, but will take much longer to calculate, and given current hardware capabilities, will be impossible to perform in real-time. The output from animation in this mode must be recorded to some media which could later be viewed in real-time.

Another required pull-down menu will host the editing functions. Such popular commands as undo, cut, copy and paste should be made available. Any of these edit commands should be made usable during any portion of the animation

process.

The two other required pull-down menus will be for attribute and object selection. Through these windows, the characteristics of the individual attributes and objects of interest could be selected. Attributes, objects and the characteristics thereof will all be detailed in chapter five.

4.6. Hardware Requirements

Current window systems implementations have exhibited a definite tendency to require a large degree of hardware capacity. It would be unrealistic to assume that any future implementation of this animation system will require any less power and functionality. Therefore, the following inventory of hardware should seem reasonable and will most likely be necessary. It should also serve as a basis for further hardware requirements.

First of all, a CPU of adequate performance must be selected. Much of the functionality of this interface calls for the real-time display of objects with hidden line and surface removal. The CPU must have access to a significant amount of primary memory, also, since the models to be animated and the algorithms involved will be complex. Next, a high-resolution monitor must be chosen. The fine adjustments that animators must make to models necessitate the

viewing of small differences in the position and orientation of the models. The monitor must have resolution fine enough to easily facilitate the viewing of the possibly many windows of interest. Also, it should have color capability to accurately designate and animate the colors of each manipulatable model.

A keyboard is required to aid in naming. Many of the objects the animator must contend with can be utilized much more efficiently by their names.

Also, a large capacity, secondary storage device is required. Animatable models, simple motions and even whole sequences of animation will be storable and retrievable to ease the burden on the animator.

Lastly, a special input device must be selected and incorporated into the computer system. Since the animator manipulates three-dimensional models in a three-dimensional space, that paradigm is incorporated into these recommendations. The input device must be able to select a point out of the three-dimensional space with a high degree of precision. Such selection devices are currently available today, but a simple substitution might be the combination of a mouse and a paddle with no end-stops. The mouse would then be used for selection in the customary two dimensions while the paddle would be used to indicate depth.

5. Creating Actors

Animators who work in the three-dimensional medium do so with the use of movable models. These models can be repeatedly adjusted in even the slightest of fashions and filmed to create motion of any detail or precision. This chapter is dedicated to the description of how animatable models can be created using this proposed interface.

The term **actor** has seen much popularity with existing animation systems [7,9,17], and so it is adopted for use here. An actor in a computer animation system is the equivalent of a model in the actual medium of three-dimensional animation. Like a manipulatable model, an actor can be given motion directions to follow. These directions will be given in both the actor and scene windows. When an actor is selected in the scene window, its image is transferred to the actor window where it can be manipulated. All changes performed on an actor's image in the actor window will be reflected on the image in the scene window. On the other hand, during the process of actor creation, its position could be typically set at the origin of the cuing model of the scene window. Once an actor is in the scene, it will be a candidate for manipulating and photographing.

An actor will be, simply, an internal, structured representation of a movable model by the computer. Such a

representation scheme is offered by the new proposed, hierarchical, graphical standard, PHIGS [25]. Using PHIGS, an actor can be composed of one or more interconnected objects. The more objects that will be included in the definition of an actor, the greater the detail and precision that will be possible in describing the motion for that actor.

An **object** will simply be an instance of one of several different **classes** of easily defined shapes. These will include geometric, parametric, photometric, photographic, and other, specialized classes of object. By connecting objects, an actor will be formed. Objects of different classes will have different characteristics, but also common ones. These characteristics will be categorized into a broad group of object-defining qualities, collectively known as attributes.

An **attribute** will be a defining quality for an object. Attributes will be grouped into two distinct sets, those that are applicable for all object types, and those that will only apply for certain classes of object. Attributes will describe how individual objects appear and behave.

A thorough discussion of the attributes that should be made available follows in the next section. Following that will be overview of the description and usage of objects. Detailed information on the editing of actors will be given in the last section of this chapter.

5.1. Attributes

This section is organized by presenting a list of common attributes that will apply for all object classes. Each attribute is described, and a proposed method for selecting attribute values will be given. Certain attributes will be omitted from this list. These attributes are concerned with only a single class of object and will be covered with the discussion of that class. This is done, because in all cases certain information about the class is needed to grasp the full import of the attribute.

Unless specified otherwise, an object must first be selected before any attributes can be applied to it. This should be done by simply pointing to it and pressing the button on the selection device. Once an object is selected, a surrounding selection area must be shown, as is the typical manner on existing windows implementations. Also, any number of objects can be selected at a given time. This can be accomplished by defining an area of selection around the objects.

5.1.1. Object Type

This attribute of an object will define its type, which could also be termed its shape. Among the geometric and parametric classes, several different types of object exist.

This attribute will be included to differentiate between the individual types. Within the discussion of the classes of object in the next section is a list of object types.

This proposed selection method will be rather simple and straight forward. Through an option in the attribute menu discussed in the previous section, an icon list of different object types should be presented to the user, so that a selection can be made by pointing (Fig 5.1.). Once an object type has been selected, an image of the object will be sent to the actor window where further attributes could be selected for it.

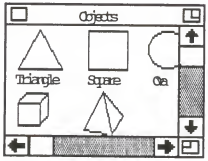

Fig. 5.1: The Object Selection Window

5.1.2. Size

The benefit of using objects of different sizes will be of great value, and such an option must be made available. A method of sizing objects, by which handles on a surrounding selection area are chosen and adjusted (Fig. 5.2,) is prevalent in current window implementations, and is appli-

cable here. A ruler should be made available to ease the burden on proportionalizing the sizes of different shapes. Object solids would have to be rotated to shape all three dimensions.

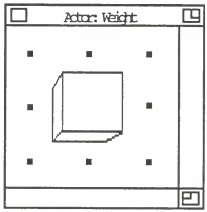

Fig. 5.2: Sizing an Object

5.1.3. Color

An object can be assigned a particular color value. The are several systems that have been designed for the selection of color [20]. Among these are RGB, HSV, HLS and CNV, the most popular being HSV since it is a more natural expression for the artist [2].

The proposed method for selecting colors uses the HSV system and is one that should be intuitive to the user and analogous to current color selection techniques. Figure 5.3 represents the color selection window that should be made available when the appropriate option is chosen from the

attributes menu. This window will include two sections. The first is the color hexagon as detailed in [21], which will be used to display all of the combinations of selectable colors, with some predetermined precision, which are characterized by having a value of one only. The user could use a pointing device to select the desired hue and

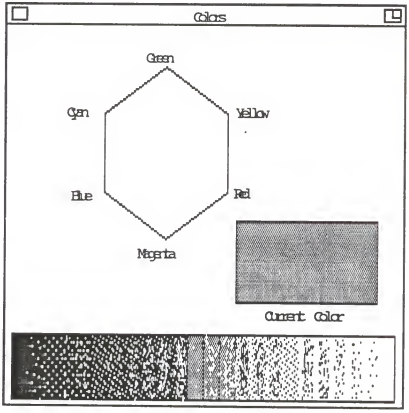

Fig. 5.3: The Color Selection Window

saturation. Once this is done, the second part of the color selection window, the value bar, will be updated to represent all of the possible colors, again with some predeter-

mined precision, that are selectable by adjusting the value parameter. The left side of the bar will always be black, and the selected shade will be on the right. The color that is selected off the value bar will be given to the selected object or group of objects. At all times a box will be shaded with the current color.

If the color capabilities of a particular hardware system are unfortunately limited, then obviously a simpler selection scheme can be chosen.

5.1.4. Pattern

An object that is shaded with only one color, can be a very limiting factor on the scope of creativity desired by an animator. Also, perhaps color will not be what is desired for an object, rather stripes or dots, for instance. If selecting a color for an object is not sufficient, then the artist should be able to select a pattern. A pattern will simply be a predetermined matrix of colored points. The pattern will be repeated over the entire expanse of any particular object selected.

The selection of patterns should proceed analogously to the selection of object types. An option from the attribute menu should reveal a window containing available patterns to choose from (Fig. 5.4). Several standard patterns should be made available to the user. The user will select a pattern

using a pointing device, but if the pattern of choice is not made available, the user must be allowed to create it using a pattern editor.

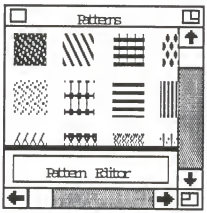

Fig. 5.4: The Pattern Selection Window

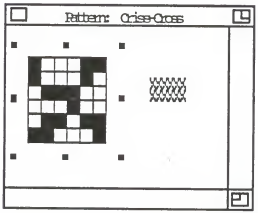

Fig. 5.5: The Pattern Editor

Used in conjunction with the Color Selection Window of Fig. 5.3.

This pattern editor (Fig. 5.5) should allow the user to select colors as mentioned above, and place those colors

into a pattern matrix. Not only will the user have control over the color coding of the pattern, but also over the size of the pattern. A pattern of larger size will seem more natural when viewed on an object after being fully image-rendered.

5.1.5. Texture

Texture can be considered a more complex variation on a pattern. As colors will be defined for individual points in a pattern, a texture will also contain attributes for displacement and reflectivity. This is the scheme adopted by Csuri, et al. in [16]. Since a displacement factor is included, the textures can also appear to be three-dimensional after full image- rendering, even though they are only defined in two dimensions.

Selection of a texture for an object should proceed similarly as for the selection of patterns. If a desired texture is not made available, then a texture editor, identical to the pattern editor but for the selection of displacement and reflectivity attributes, must be made available.

Displacement values can be selected using the following scheme (Fig. 5.6). A series of line segments can be displayed, each slightly longer than the next. Each of these line segments will be selectable and will refer to the dis-

placement value for any of a number of selectable points in the texture matrix. A small, selectable, movable pointer should be included to denote the current displacement value. Also, a numerical input box should be included to enter the displacement value directly. Regardless of the method of selection, all areas involved in displacement selection should be dynamically updated. The selection of reflectivity values proceeds below.

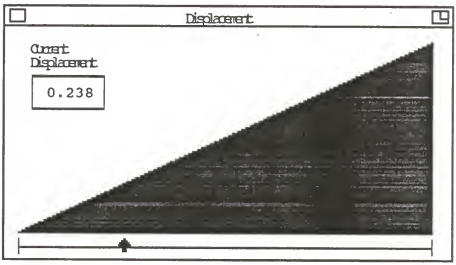

Fig. 5.6: The Displacement Selection Window

5.1.6. Reflectivity

Reflectivity can be termed the degree at which light is bounced after it has hit an object. An object can have a reflective value between zero and one inclusive. A reflectivity of one designates that all light is reflective and the object will be seen vividly, perhaps with the image of

other objects reflected in it. If a zero value is chosen, however, the object will appear black. This attribute is strictly an image rendering attribute and it, like the other attributes of this class discussed below, can only be viewed during the fully image rendered recording option that will be offered in the file menu.

The selection of all image rendering attributes will be similar. To choose a value for one of these attributes, a window must be made available after choosing the appropriate option from the attributes menu. This window (Fig. 5.7) will contain all of the selection areas for the three image rendering attributes. For each attribute, a theoretical surface will be displayed upon which rays are shown with the different attribute characteristic which denotes the degree of the appropriate attribute. This degree will be represented, in each case, pictorially upon the surface and also numerically. The pictorial display will be accompanied by a manipulatable scroll bar. Either the scroll bar or the numerical representation could be selected, and a new value for the attribute could be chosen respectively by scrolling the percentage greater or lesser, or by simply typing in the new value. Regardless of which is chosen, all areas of the screen are simultaneously updated.

For reflectivity, the pictorial representation will show the amount of light reflectivity by proportional line

thicknesses. The greater the reflectivity, the thicker the reflected ray line will be.

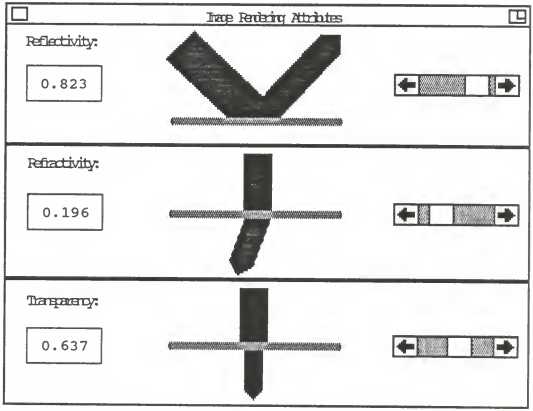

Fig. 5.7: The Image Rendering Attributes Selection Window

5.1.7. Refractivity

The degree to which light bends after traveling through a surface is the degree of refractivity. An object can have a refractive value between zero and one inclusive. A refractivity of one designates that the light is bent to parallel along the surface. If a zero value is chosen, the original path of the light before encountering the object

will not be altered after having passed through it. This is another image rendering attribute, and as such, can only be viewed during the fully image rendered recording option that will be offered in the file menu.

The selection scheme for refractivity is also offered in the image rendering attribute window in Fig. 5.7. For refractivity, the pictorial representation will show the amount of light refracted by a drawn vector showing the path of the light after passing through the surface.

5.1.8. Transparency

The degree to which light is admitted through a surface is the degree of transparency. An object can have a transparent value between zero and one inclusive. A transparency of zero designates that no light is allowed to travel through the surface; it is totally opaque. If a one value is chosen, then all light hitting the surface is transmitted through it; barring other attributes such as color, pattern and texture, the object would be invisible.

The selection scheme for transparency will also be offered in the image rendering attribute window in Fig. 5.7. For transparency, the pictorial representation will show the amount of light transmitted by proportional line thicknesses. The greater the transparency, the thicker the transmitted ray line will be.

5.1.9. Other Attributes

The above list contains the recommended minimum of the possibly vast number of attributes that can be assigned to objects. Special attributes of any kind could be added to the list, particularly other image rendering attributes. The technology of rendering images is growing rapidly both in power and in scope, and doubtlessly, some algorithms may require additional object attributes.

There are two innate attributes that every object has: position and orientation. These attributes will not be selected nor manipulated like the others and so will not be included in the same list. These attributes are intimately and logically linked to the actor and scene windows, and so are detailed in the next chapter.

Also, there are a number of object classes which require unique attributes. These are introduced during the discourse on the individual object classes.

5.2. Objects

This section details the different classes of object that should be made available in a future implementation. Because of the large numbers of widely disparate object classes, this is by no means an exhaustive list. Rather, it is open-ended, so that object types and even object classes

can be added.

5.2.1. Geometric Object Class

This class will include the simple, easily-defined objects of plane geometry. These will include triangles, squares, rectangles, circles and regular polyhedra. Also included in this list will be the regular three-dimensional counterparts of each of these: pyramids, cubes, spheres, and so on. Even more complex, ruled-surface shapes should be included in this list, such as cylinders and cones. Finally, this class of object could be further expanded by including such objects as parabolic cones, revolution surfaces, and swept volumes. All of the simple object types have been used successfully in existing animation systems [6-9,19], as well as the more complex shapes [9].

All of the attributes mentioned in the above section will be relevant in describing the appearance of any object of this class.

5.2.2. Parametric Object Class

This class will include those objects which must be defined using a series, or matrix, of control points. They are easily more complex than objects of the geometric class, but they will also potentially lend a degree of freedom to

the creativity of an artist. These objects include polygon meshes such as those used in [19], and the Bezier surfaces and beta-splines offered in [9]. There are several more examples of usable parametric objects and they may be included in a future implementation, if desired.

5.2.2.1. Control Point Attribute for Parametric Objects

The control points of a parametric surface define the shape of the surface. They must be defined in a three-dimensional space, so the cuing model of the scene window must also be incorporated in the window for selecting control points (Fig. 5.8). This window should be made available after the parametric object is selected from the object type select window. Once a predetermined sufficient number

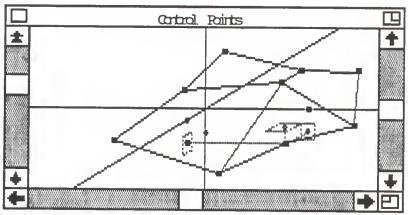

Fig. 5.8: The Control Point Selection Window

of control points have been designated, then the polygon mesh should be drawn in line form to be editable, also in

the fashion of the scene window. Each control point should be made selectable and movable.

5.2.3. Photometric Object Class

The objects of this class will be the logical light-emitters used in conjunction with the image rendering algorithms to produce simple or complex lighting effects. The possibility of including any number of these objects in a scene should be made available, and they should be manipulated identically as with geometric objects.

These lighted objects are defined to have volume, so that more complex effects are possible. Each light should also have the attributes of size and color. In addition to these, two other, unique attributes will be required to define a photometric object: intensity and diffusion.

5.2.3.1 Intensity Attribute for Photometric Objects

Intensity can be termed the degree to which the luminosity of a light source is not dampered. A photometric object could have an intensity value between zero and one, inclusive. An intensity of one will designate that the maximum luminosity will be emitted from the light source, the maximum being at some predetermined level. If a zero value is chosen, then the light source will not emit any

light. The degree of intensity of a light source will only be viewable during the fully image rendered recording option that will be offered in the file menu.

To choose a value for intensity, a window must be made available after choosing the appropriate option from the attributes menu. This window (Fig. 5.9) will contain both the selection areas for intensity and for diffusion. The area for intensity selection will include an icon image of a light source, with variable length line segments representing light rays and the current degree of intensity. A scroll bar should be made available to dynamically adjust the current intensity level, and a numerical entry box should also be made available to enter the value directly. Regardless of which selection device is chosen, all areas of this window are to be updated dynamically.

5.2.3.2. Diffusion Attribute for Photometric Objects

Diffusion can be termed the degree at which the light disperses from the initial point of the light source. A photometric object could have a diffusion value between zero and one, inclusive. A diffusion of one will designate that the light disperses equally to all points in a sphere surrounding the source. If a zero value is chosen, then the light rays will all be parallel with no distance between them; a laser effect. The degree of diffusion of a light

source will only be viewable during the fully image rendered recording option that will be offered in the file menu.

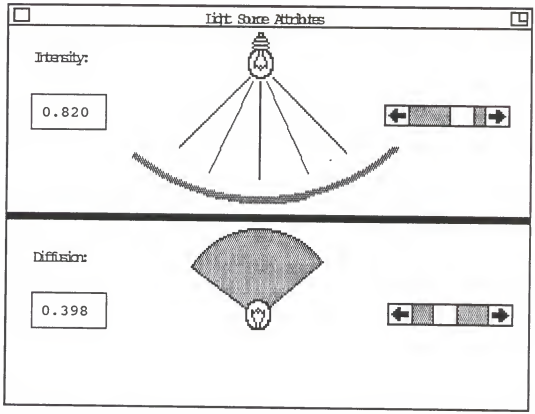

Fig. 5.9: The Light Source Attributes Selection Window

To choose a value for diffusion, a window must be made available after choosing the appropriate option from the attributes menu. The diffusion selection area of this window (Fig. 5.9) will contain an icon image of a light source, with differing numbers of line segments, dispersed about the source, representing light rays and the current degree of diffusion. A scroll bar should be made available to dynami-

cally adjust the current intensity level, and a numerical entry box should also be made available to enter the value directly. Again, regardless of which selection device is chosen, all areas of this window are to be updated dynamically.

Since diffusion will assume that the emitted light rays have a direction, the orientation attributes of the photometric object will be very important. When orienting one of these objects, a vector indicating the direction of the light should be temporarily viewable.

5.2.3.3. Ambient Light

There will exist one more general type of photometric object which defines the ambient light characteristics for an entire scene. This ambient light will be the light that is in evidence that is not attributable to any given light source, and so cannot be attached to a photometric object. Rather, the attributes for ambient light will be included in the scene, itself. The only attributes that will be relevant for ambient light are color and intensity.

5.2.4. Photographic Object Class

This class of object will include the cameras that are used to photograph the scene. Since any number of different

perspectives of a scene will be possible, the capability of creating and manipulating any number of these cameras should be made available. Each camera has its very own view window. Since there may be a large number of cameras viewing a single scene, not every view window must be viewable at all times. However, at all times at least one of the cameras and its associated view window, will be designated to be currently active by the global view window, and should be made appear on the monitor.

These cameras are logical cameras, only, and occupy no volume. Each camera will be defined to be only a single point. However, since cameras must be selectable and since a point is not selectable, each camera must be given a name. When a camera must be selected, the following practice should be used. An appropriate option from the objects menu can be selected which displays all of the names of the active cameras. The camera of interest can be selected by choosing its name from this list. While a camera is selected, a small camera icon and its view pyramid (Fig. 5.10) will appear in the scene window to designate its position. At the same time, the icon will appear in the actor window. This icon can then be oriented and positioned just like any other object.

5.2.4.1. Focal and Magnification Attributes for Photographic Objects

A camera's view pyramid will be used to designate the focal and magnification settings of the camera. The view pyramid (Fig. 5.10) will be a dotted-line, rectangular pyramidal shaped area, with the apex at the camera. Along its length, from the center of the base to the apex will be another dotted line. The heighth of the cone will represent the focus. Those objects at the camera's focus will be seen clearly, while those nearer or farther away will gradually

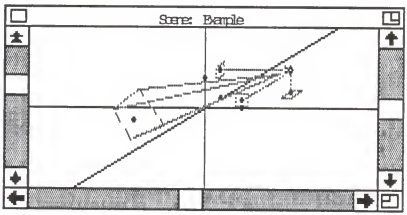

Fig. 5.10: Focus and Magnification Selection

blur. The magnification will be represented by the size of the base of the pyramid. All of the area within the pyramid base will be what is shown in the camera's view window, and subsequently, on the recorded medium for viewing. The larger the base, the more information that must be placed inside the view window, so the image appear smaller. If the

size of the view pyramid's base were reduced, then the image would be appropriately magnified. Also, since the base of the view pyramid is rectangular and not square, each dimension could independently manipulatable for special effects.

The size of the pyramid base as well as its height are both selectable and manipulatable. To adjust the focal length, the base-end of the center line mentioned above may be selected and move either closer to the camera, or farther away. Motion of this point could not only be strictly linear, but should also be movable to any point in the scene. This could not only be the fashion to adjusting a camera's focus, but for orienting it as well. The magnification could be adjusted by manipulating handles around the base of the view pyramid. These should be manipulatable just as a normal rectangular geometric object. As an added special effect proposal, it mustn't necessarily be the case that the middle line be at right angles to the base plane of the view pyramid. For effect, part of the view could be closer, the rest farther away. Regardless of the form in which camera animation is utilized, all changes performed on a camera in the scene or actor windows will be automatically updated and represented in that camera's view window.

To facilitate the usage of this camera of fair complexity, the lens and aperture model described by Potmesil and Chakravarty in [13] could be used.

5.2.4.2. Orientation Attribute for Photographic Objects

In addition to the orientation possibilities offered by the view pyramid, the camera should also be orientable in the normal fashion like other objects. Another orientation tool might also be for linking the base-end point of a camera to a given object on an actor. This would make the camera automatically follow that particular object throughout an animation sequence. To do this, an appropriate option from the object menu could be selected along with a camera and that camera's object of interest.

5.2.4.3. Filter Attribute for Photographic Objects

The choice of a filter for any given camera must also be made available for this interface. Filters will add special effects to camera views and have been successfully utilized in existing animation systems, such as [9,12]. A camera could have zero or more of these filters logically in front of the camera lens at any time. Some of the filters used in [12] are mentioned below:

A color additive, subtractive and replacement filters could be used to modify the colors viewed through a camera. A fog filter could be used to make a scene dim and hazy. A stereoscopic filter films a scene in two, slightly different perspectives, one in cyan, the other in red. When the scene

is viewed through special glasses, it appears three-dimensional. Another special filter that could be used is a matte. A matte blocks part of a scene from view. Depending upon the shape of the matte, particular special effects may be achieved.

Regardless of the type of filter required, a list of filters (Fig 5.11) must be made available, through an attribute menu option, so that one or more may be chosen. Also, an editor must be made available to custom design mattes of any shape. This editor would be very similar to that used to create patterns.

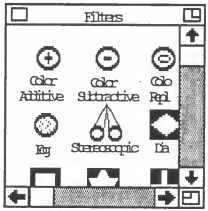

Fig. 5.11: The Lens and Filter Selection Window

5.2.5. Frame Object Class

Frame objects are two-dimensional, bit-mapped images. These have successfully been incorporated into [9] and have the potential for introducing several special effects.

Frame objects must be manipulatable identically as with other object classes. The only attributes that are applicable to frame objects, however, are size and the image rendering attributes: reflectivity, refractivity and transparency.

5.3. Actors

This section details how a future implementation should allow an animator to manipulate the three-dimensional models, actors.

5.3.1. Linking Objects

Once the desired objects have been selected, they must be logically connected together to form actors of varying complexity. Any object of any class must be connectable to any other, even if the two classes are not the same. This should be done in the following manner. First, a point on each of the two objects must be selected. If a photographic object is being used, there will exist only one choice for a selectable point since these objects have been defined to be points, themselves. If a three-dimensional solid object is being used, then the point must be selectable from anywhere inside the volume or on the surface of that object. To facilitate this, the three-dimensional cuing model could

again be used, this time with a superimposed image of the solid object to visually indicate the animator of the exact point of interest (Fig. 5.12). This cuing model would include highlighted points on each axis of the intersection of the axis with the solid. If a parametric object is being selected, since they are mathematically complex, perhaps only the control points could be selectable as points of connection, again using the cuing model.

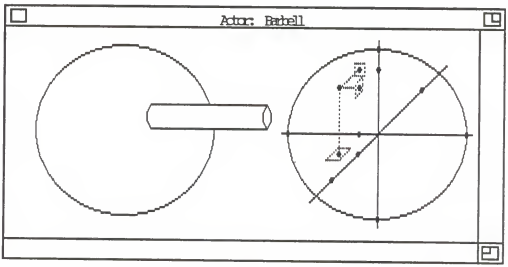

Fig. 5.12: Three-Dimensional Solid Object
Connection Point Selection

Once points have been selected on the two objects, an appropriate item from the objects pull-down menu will be offered to logically connect the two objects at their respective, chosen points. Connecting the objects, however, will not be the only task at hand in building actors. As

another step, the two objects must be oriented relative to each other. This orientation could occur either before the connection or after. During the animation process, relative object orientations will be made possible, and will be discussed further in chapter six.

5.3.2. Deleting Objects

Occasionally, the animation will require the removal of objects from an actor. This option must be made available. The actions to perform this operation should include selecting the object while the associated actor is in the actor window and then using the familiar cutting tool from the edit menu to remove it.

5.3.3. Positioning Actors

Once all of the objects comprising an actor have been assembled in the actor window, the actor must still be positioned in the scene window before it can be viewed through a camera. While an actor is being built for the first time, a representation in the scene window of the actor will be visible, but at the origin of the scene window's cuing model. If an actor is being edited, those edits will also be reflected on the actor's image in the scene window, but wherever the actor was located before being selected. Re-

gardless of where the actor exists in the scene window, it can be repositioned. This should be done simply by selecting the actor in the scene window and, using the cuing model, place it somewhere else. The same orientation parameters will remain in effect, however.

5.3.4. Splitting Actors

At any time, objects comprising an actor could be separated. This would be done simply by selecting a particular object of interest and using an appropriate object menu option to destroy one of the connection links it will have with other objects. Once this is done, two actors will exist where only one was. Since only one actor can exist in the actor window at a time, the actor which contains the selected object will remain and the other will be removed, unselected, from the actor window. Any objects connected to the object of interest will still retain the connection links. If that situation will be that the two new actors still have at least one link after the first has been severed, then two new actors are not created, until all have been removed.

5.3.5. Joining Actors

Two actors or objects should also be joinable. To join

two actors, the same procedure should be taken as with joining two objects. One object of each actor should be selected, and then linked. Once this has occurred, one actor will exist where two did previously. Since only one actor can exists in the actor window at a time, however, object selection must occur separately, and the link process must be done via an object menu selection.

5.3.6. Deleting Actors

Occasionally, the animation will require that an actor leave the scene. To remove an actor from the scene, it will be simply selected from the scene window and cut. This is a similar operation to removing objects from an actor.

5.3.7. Saving/Retrieving Actors

All actors must be savable on external storage. This will allow both for the retention of complex actor parameters and for the duplication of actors. When an object is created, oriented, positioned or deleted from a scene, it must be savable. When an object is saved, all information pertaining the actor, including objects used, connection points, relative and absolute orientations, actor position and all current attribute values, will be saved with it, so that it may be retrieved in an intuitive manner, identical

to the form it held when last saved.

To save or retrieve an actor, the appropriate option should be made available in the filing menu.

6. Animating Actors

Now that the descriptions of actor creation and modification have been given, the detailings towards the animation of the actors will proceed in this chapter. A description specifying the proposed usage and functionality of each of the four main windows will be given first. Following that, recommendations towards the animation of attributes, objects and actors will be presented.

6.1. Window Functionality

This section establishes the main functionality and proposed usage of each of the four main windows: the scene, actor, view and track windows. For a description of the appearance of each of these windows, refer to chapter four.

6.1.1. Scene Window

As mentioned before, the scene window will give the entire view of the area that can be photographed using a logical camera. Two main functions of the scene window will be in evidence, not only to view the entire animatable area, but also to select and position individual actors.

Since the scene window could accrue much detail during a complex animation sequence, opportunities for viewing only

a portion of the window, but at a closer magnification, will be given. This will be performed by using the third dimensional scroll bar located on the left side of the window. By moving completely away from the scene, the entire scene will be made visible. Also, by moving closer, and perhaps adjusting the other two dimensional scroll bars, close up images of actors can be perceived.

To facilitate animation requiring the motion of actors through space, the actor must first be selected and then be repositioned in the scene window. A selected actor will be easy to locate, since it will be the one denoted by the planar and axial cues of the scene window. To select an actor, one of the following two schemes could be used. The first option would be to use only a two-dimensional pointer. This will make the selection of an actor simpler, but only in those cases where the actor of interest will not be partially or even totally obscured by another. If this is the case, the more complex method should be used. This would require the usage of a three-dimensional pointing device. As the pointer moves deeper into the scene, those actors that are visible closer to the viewer would be temporarily removed or dimmed so that a clear image of the desired actor can be eventually perceived and then selected.

6.1.2. Actor Window

The actor window will be used to gain immediate proximity to the detail of an actor so that it can be easily changed or manipulated. There will be two main factors involved in the functionality of this window. In addition to providing a detailed image of an actor, it will offer an avenue towards the selection and orientation of both the individual objects and the actor itself.

When an actor is selected in the scene window, its image will be immediately represented in the actor window. At this point, the entire actor will be selected, not an individual object. To select an object, one of the following two methods could be used. If the object is immediately visible in the actor window, a simple two-dimensional pointing device could be used very efficiently. If, on the other hand, the object is either partially or totally obscured by other objects, then the actor will have to be temporarily rotated until the object of interest comes into view, where it could then selected. Once the object is selected, the actor could be returned to its previous orientation. To visually locate the selected object, object handles should be provided. These handles will also add functionality to the manipulation of objects, and will be discussed further, below.

Once an object is selected it can be oriented. To do

this, the following method should be used. The current size of the window will be defined to be the area of orientation denotation. If a pointing device is dragged from the left side of the window to the right, that will denote a 360 degree right rotation about the Y-axis. If the pointing device is dragged from the top side of the window to the bottom, that will denote a 360 degree downward rotation about the X-axis. The third dimensional pointing coordinate would then be used to denote rotations about the Z-axis. By not dragging the mouse completely across the window, partial rotations can be achieved. Also, more complex rotations are possible by dragging in diagonal directions. Whatever degree or direction used, however, the selected object will mimic the desired rotation.

Because of the logical model used in defining actors, when one object is oriented, it will be a local orientation and it will not affect other objects. This will be different than when the actor is oriented. This global orientation will affect all of the objects of the actor. To orient an actor, first no object can be selected. This means that the actor itself will be chosen. The actor should be oriented in precisely the same manner as are the individual objects.

6.1.3. View Window

The view windows will be used to see images that will be of interest by the logical cameras. Since they are a view only, they don't have, themselves, any specific functionality. Neither actors nor objects should be selectable through a logical camera, since this is not an intuitive process.

6.1.4. Track Window

The purpose of the track window will be to pictorially denote the time frames in which different tracks of animation occur. To facilitate this purpose, several aspects of functionality will arise in this window. With this interface, should the animator perform any action to create, edit or destroy an attribute, object or actor, the case must exist that the action could be given a time span. Simply put, this will allow the actions to be seen. Much of the functionality of this window, therefore, will be dedicated to the denotation of time spans. The rest of the functionality will be dedicated to the specification of the current time frame.

Any action performed to an attribute, object or actor could be linked to one or more time frames. The last create, edit or delete action will be the candidate to which a

time could be given. To do this, the pointing device could be used to select a time frame, and then dragged to another time frame. The span between the two will be the track specifying that particular action, and will be the time allotted to perform it.

Any track is editable. First the track must be selected using a simple pointing device. The start and finish end-points of the track could then be adjusted forwards or backwards to expand or contract the time span of the given animation. Also, any track could be selected and then deleted by simply using the cutting tool from the edit menu.

A track could be of any size, but there will be at any time a finite, current number of frames. Should a track be dragged past the last frame, additional logical frames must be created to accommodate the track, and thus a finished product, of any size. At the beginning of the animation process, only one frame will exist in the track window. No track can be dragged before this first frame.

Many animation systems [6,7,9,14,19] have tackled the difficult concept of synchronization. This aspect is incorporated into these recommendations in an intuitive and implicit manner. If two actions need to start together, their associated tracks would simply be started on the same frame, and so on.

The other functionality aspect of the track window will

be that concerned with the current time. The frame that will be designating the current time will be shaded with some scale of grey. To select a current time, the user should be able to quickly press twice the pointing device button on a frame. The view from every window should be updated to represent the state of the animation sequence at this new current time frame.

6.1.4.1. Mathematical Interpolation

Since one or more frames may be designated the time span for a particular animation, a scheme must be selected to mathematically interpolate the in-between phases of the transformation. The longer a track will be, the more in-between images there will exist to interpolate. Should a track be only one frame long, the animation will be instantaneous, and no interpolation would be necessary. There do exist, however, a multitude of interpolation schemes as mentioned in [19]. Each track should default to linear interpolation, since this is the most obvious. Other interpolation schemes, though, should be selectable to provide more complex animation without specifying that complexity manually. Schemes such as ease-in and ease-out [19] should be selectable from a pull-down menu. Other schemes should also be offered, and perhaps an interpolation editor should be provided, though will probably not be necessary except in

extreme cases.

6.2. Attribute Animation

This section details specifications on the implementation for object attribute animation. For each attribute, a discussion concerning the selection of an animatable transformation will first be given. Then, several examples will be introduced, including citations of capabilities that should be included in a future implementation that are not obvious.

6.2.1 Object Type

Perhaps the potentially most complex animation interpolation domain that can be included in a future implementation would be that concerned with calculating in-between phases for object type transformations, yet this capability must exist. To specify an object type transformation, one object must be selected from the actor window, and then another selected from the attribute pull-down menu. The new object type will replace the old on the actor, and then the object type transformation could be linked to a track.

Not only should the system be able to mathematically interpolate between two and three dimensional geometric objects, but also between other object classes as well.

Some of the more complex examples of this type of transformation might include parametric objects, whose control points dictate the phases of the interpolation. Converting an object to and from a photometric object would also seem complex. Not only must the physical aspects of the object change, but also must the intensity and diffusion attributes adjust mathematically. In addition, converting to and from a frame object would entail, among other things, projection calculations. The one object class that should be exempt from transformation would be that for photographic objects. This is because every logical camera must have its own view window, and the potential complexity in managing a large and dynamically varying number of view windows would be formidable.

6.2.2. Size

Contrary to the interpolation for object type, that for object size would seem the most simple. A selected object's handles could simply be readjusted to represent a new size, and then the transformation could be linked to a track.

6.2.3. Color

The transformations to change an objects color should also be quite simple. To change an object's color, the

object must first be selected, and then a new color chosen from the attribute menu. Once this has occurred, the color change could be linked to an animation track.

The color transformations would probably best be accomplished at the lowest level, which would be the individual red, green and blue components. Each of these components would be mathematically interpolated in parallel to achieve the effect. The capability of giving a separate track to each of the three components of color is not recommended for two reasons. First, it would be an unduly complex, non-intuitive approach, and second, if the animator desires to animate a single component only, then an intuitive approach would be to select a target color that appears to have the other two color components the same.

6.2.4. Pattern

To interpolate between two different patterns, the in-between color of each point must be determined individually. The specification for pattern animation should be identical to that for color.

6.2.5. Texture

For texture, not only should the color attribute of each point be individually interpolated, but also the dis-

placement and reflectivity attributes. To specify texture animation, the same procedure as that used for colors should be used.

6.2.6. Image Rendering Attributes

The image rendering attributes: reflectivity, refractivity and transparency, could be manipulated to reflect animated change in the image quality of different objects. To specify an image rendering attribute change, first the object must be selected, and then new values for these attributes could to be selected. Once this has occurred, the animation could be linked to a track.

Interesting effects, such as fade-in, fade-out and distortion are possible by adjusting one or more image rendering attributes simultaneously.

6.2.7. Parametric Object Class Attribute: Control Points

To animate the shape of a parametric object, the control points which define its shape could be manipulated. To do this, the parametric object must be selected, and then the control points could be edited in the same fashion as they were created. When a satisfactory new shape is achieved, the change could be linked to a track.

6.2.8. Photometric Object Class Attributes

To animate the characteristics of the lighting model, the specific photometric object class attributes, diffusion and intensity, could be manipulated. This would be done very simply by selecting the light source from the scene, and then modifying the two attributes. After new values have been selected, the changes could be linked to an animatable track. The color of the light source, as well as its shape, should also be animatable in the fashion described above.

6.2.9. Photographic Object Class Attributes

The attributes of a photographic object are also animatable. Each of the following examples first assume that a logical camera was selected, and will be currently visible in the scene window. After each of the changes have occurred, the animation could be linked to a track.

First, the focus and magnification of a particular logical camera can be adjusted using the method specified in the previous chapter. Second, while the theoretical camera image is in the actor window, its orientation in all three dimensions could be modified. Third, a camera's filter could be mathematically interpolated into a filter of another sort simply by selecting a new filter and then linking

the change to a track.

There will be two other types of possible animation concerned with the photographic objects that will not be performed upon an individual camera. First, to select a new current camera, either the label field of the global view window could be selected and changed, or a menu option could be used. Second, if the current camera change track happens to be longer than a single frame, then image fades of various speeds could be attained. Other types of special effects, such as wipes and cross dissolves as detailed in [12], could also be used as special camera effects.

6.2.10. Frame Objects

The research and literature for two-dimensional, mathematical interpolation is vast and widely varied [22,23]. There is, however, no currently adequate scheme, because of several problems as detailed in [5]. Once an acceptable method is discovered, it should be incorporated into these proposals. The only alternative available at this time would be to treat frame object interpolations as if they were simply pattern attribute animation. This would succeedingly modify each point of one frame object until it appeared as the second. This would be further complicated by the fact that the two objects may be of different sizes and orientations. Selection of frame object animation would

proceed in the natural way specified above.

6.3. Orientation Animation

All orientation motions will be described to actors and objects in the actor window, as described above. If a single object is selected, it will be the subject of the orientation, but should an actor be selected, the entire group of objects will rotate similarly. Once an object or actor has been oriented a desired amount, the orientation action could be linked to a track of any length.

6.4. Position Animation

All position motions will be directed in the scene window, also as described above. When an object is positioned, its orientation will not change. To achieve the animation effect of an actor moving in space while also moving relative to itself, the position and orientation motions given the actor would be each given synchronized tracks. At the same time, synchronized tracks detailing attribute animation could also be created.

6.4.1 Associated Motion

If a large amount of detail is to be synchronized on individual tracks, possibly amongst several objects or even

actors, an alternative to the tedious task of manually synchronizing them should be offered the animator. This alternative will be called associated motion, which is an aspect that could be given to any actor, object or attribute. A similar concept is discussed in [24]. When one of these is associated with another then it will animate automatically depending upon the current state of the latter. For instance, should the color attribute of object be associated with that of another object, then the first object's color could reflect, automatically, the second's. This will be further complicated since it may be desired that the range of values of one attribute may not directly correspond to that of another attribute.

This concept could also be expanded to include objects and actors, as well. For instance, the position of one actor could be automatically updated according to the position of another. As one more level of complexity, the prospect of associating attribute, objects or actors, not only with other, identical types, but also with other types that are not identical. A possibility would be associating the orientation of an object on one actor with the intensity of a photometric object of another actor.

To perform an association, first the guiding actor, object or attribute must be selected, and one of that type's parameters must be altered to reflect the range of motion or

change that is to be used as a guide. Second, the associated actor, object or attribute must be chosen and the range of motion or change of one of it's parameters must be similarly represented. Lastly, the association track must be created and dragged in the track window. The association will be active for as long as the track is. This whole activity would be assisted through the use of pull-down menu options.

When the guiding actor's attribute is at the low end of its range, then the guided actor's attribute will also be at the low end of its range. As the current value of the guiding actor's attribute will adjust within its range, so will a proportional, automatic adjustment be made to the guided actor's attribute value within its own range.

7. Conclusions

This paper has presented a detailed set of recommendations towards a direct manipulation interface for three-dimensional computer animation. These recommendations were derived using a functional union from literature of both of these fields. No effort was made either to provide a structured design of the proposed system, or to offer any insight pertaining to the actual implementation of such a system. Rather, the result is a complete series of ideas and concepts that could be used to create a formal, structural design and implementation for the system. The system contains the important interface features found in existing animation system which have proven beneficial to animators.

Presented below is a concise summary of the recommendations offered in this paper. Following, is a discussion on the viewed weaknesses of the system, and then a list of likely directions for future study in this area.

7.1. Recommendations

There exist many systems today that are used to produce three-dimensional computer animation. Most are good, yet remain excessively difficult for the average artist to use. The direct manipulation interface is an ideology that is used to cognitively bind normal practices to operations on

the computer, and thus makes the computer easier, friendlier to use. If ideas from both of these fields were used, a friendly computer system for developing three-dimensional computer animation could be created. This paper is the first step towards the realization of that system.

7.1.1. Use four different types of windows

The interface will involve the usage of four types of windows. Each window type will have its own purpose and functionality.

° **The Scene Window.** The scene window will be used to view the entire area upon which the animation will occur. This window includes both a three-dimensional cuing model and a third dimension scroll bar to facilitate the natural perception of the scene. It will be in the scene window that actors are positioned.

° **The Actor Window.** The actor window will be used to orient the individual actors. A separate window from the scene window should be used for two reasons: first, to reduce the information overload potential of the scene window; and second, to achieve a closer, magnified view of an individual actor to facilitate selection of the possible minute detail.

° **View Windows.** The scene window will present the view of the entire scene, but a perspective of the scene through

a logical camera will be displayed in a view window. This window has no functionality, but is useful to preview the image that will eventually be rendered to some recorded medium.

* **The Track Window.** To implement the popular and powerful tracking system of recent animation systems, the track window will be included. A track is a designated length of time in which some simple animation will occur. This window will pictorially display each individual track and its relative times of animation.

7.1.2. Use a hierarchically-defined animation model

The concept of the manipulatable model will best be incorporated using a hierarchically-defined animatable model. This hierarchy is illustrated in Fig. 7.1.

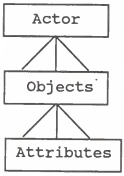

Fig. 7.1: The Animatable Model Hierarchy

* **Actors.** An actor is the computer equivalent of an animator's model. Each actor is created by logically con-

necting one or more objects together.

° **Objects.** An object is a member of one of a series of classes previously defined in the literature. These classes include geometric, parametric, photometric, photographic and frame objects. Geometric objects are the simple shapes, their three-dimensional equivalents and other shapes defined by ruled surfaces, and even more complex surfaces. Parametric objects are those characterized by the use of control points which include polygon meshes, surface patches and splines. A lighting model is implemented using any number of photometric objects which are only used in the image rendering function of the proposed system. Logical cameras are used to produce images of the animating scene are the objects included in the photographic object class. Lastly, frame objects are simply two-dimensional bit maps of the classic variety.

° **Attributes.** Every object has a set of defining qualities called attributes. A number of common attributes will be used for most object classes. These include object type, size, color, pattern, texture, and those for image rendering. Some individual object classes, as well, will have their own attribute types which will be used to accurately and creatively define them.

7.1.3. Use the direct manipulation interface for actor creation

Paradigms for the selection of objects and attributes for those objects have been given, and are be consistent with the direct manipulation interface. The selection of attributes and of objects will be performed in a natural fashion. In addition, complete control over the connection of objects has been defined, including the possibilities of connection points in three dimensions.

7.1.4. Use the direct manipulation interface for actor animation

Paradigms for the animation of actors, objects and individual attributes have been offered. The explicit control of orientation motion in the actor window, and position motion in the scene window has been described. To animate actors, objects or attributes, paradigms towards the linkage of actions to the tracks in the track window have also been given.

7.2. Weaknesses

There are a number of perceived weaknesses in this set of recommendations. Many of these arise from limitations of current hardware capabilities or of the direct manipulation concept. None of these, however, should be sufficient to

significantly limit the scope and power of the proposed system.

First of all, these recommendations do not fully attain 100% cognitive binding. It is simply not possible, under current hardware capabilities, to provide a view of a scene, or even a perspective in perfect WYSIWYG. The requirements of image-rendering algorithms are too stringent now to expect fully rendered images in real-time.

One of the areas which is not touched upon in this paper is that concerning the sound track. The sound track is audible portion of an animation sequence. It is usually created before any of the animation, so that motion can be synchronized with it. A few notes about introducing a sound track into an interactive animation system are presented in [5]. The sound track was not included in this paper, because at all times the interface calls for the visual display of the animation sequence frozen in time. It would be extremely ponderous to synchronize these frozen images of the animation with sound, since sampled sound, at a given time, is a non-discernable input. This remains a research topic at this time.

Another perceived limitation of the proposed system is the procedure for linking a particular type of animation to a track. Using these recommendations, only the last animation action directly specified is a candidate to be linked.

Ideally, this is very restricting since it is improbable that all desired actions could be expressed in a simple sequential manner. Perhaps a list of previous actions could be kept and the one that is most pertinent would be linked to a track.

The last weakness is one that is involved with the entire system, and is inherent with most animation systems such as [9,19]. The designers of these systems found that some of the most complex animation sequences that an animator may wish to create will simply be too difficult to express using an interactive or direct manner. Two solutions to this problem exist. As one solution, the system designers implemented an animation programming language or syntax that could be used to describe animation concisely and exactly. Also, because of the inherent structure of procedural programming languages, animation sequences of any complexity could be written. Giloth and Veeder in [11] suggest that an artist will overcome any obstacles to be expressive in a medium of interest, even if that medium is the computer. For these reasons, the case will probably occur that an animation language will nicely complement the direct manipulation interface described in these recommendations. Attempts to describe such a language can be found elsewhere in the literature [6-9,14,17,19], and are beyond the scope of this paper.

As another but less exploited solution, an exhaustive library of motion primitives could be created for the animator to use. Motion primitives are complex sequences of animation that are difficult or impossible to accurately describe directly. An example would be human ambulatory movement which continues to be one of the most complex animations to realistically portray. These motion primitives could exist as complex sets of tracks and could be merged into the animation using simple copy and paste operations. Using motion primitives, the animator would be able to describe complex motion in a very simple fashion without necessitating programming. However, the library of motion primitives would have to be considerable to contain all of the animations an artist is likely to require.

7.3. Future Directions

The most obvious direction to be taken from this point is to provide a detailed, structured design for this proposed system. Such a design would be lengthy and encompassing. Following that would be an implementation.

A specific example of a possible enhancement to the design would be the inclusion of a wider variety of object class types and attributes, since these seem to be a very real limiting factor on the spectrum of actor that could be created.

The system should be designed to evolve. New breakthroughs in image rendering algorithms and direct manipulation paradigms are occurring regularly, and will undoubtedly introduce new ideas for incorporation.

Several ideas and questions have surfaced as to possible future implementations and uses for this interface. Since there are many image rendering products available commercially, duplicating that considerable amount of effort would be wasteful. Rather the recommendations presented in this thesis could be used to create an interface which could be a front-end for an existing rendering product. This way, the mathematical framework for each image could be computed by the interface and given as input to the image renderer. In addition, the interface could also be a pre-processor front-end to an existing animation system. The animation designated by the artist using the interface could be translated into an animation language which could be compiled using the other system. This would also greatly reduce development cost without compromising the quality of the finished animated product. The system of the interface, however, must still contain the pencil test animation mode to greatly ease the complications of editing.

BIBLIOGRAPHY

1) Crow, F. C., ''Shaded Computer Graphics in the Entertainment Industry,'' Computer, Vol. 11, No.3, Mar. 1978, pp.11-22.

2) Booth, K. S., D. H. Kochanek, M. Wein, ''Computers Animate films and Video,'' IEEE Spectrum, Vol. 20, No.2, Feb. 1983, pp. 44-51.

3) Stern, G., ''SoftCel - An Application of Raster Scan Graphics to Conventional Cel Animation,'' Computer Graphics (Proc. SIGGRAPH '79), Vol. 13, No. 2, Aug. 1979, pp. 284-288.

4) Levoy, M. ''A Color Animation system Based on the Multiplane Technique,'' Computer Graphics (Proc. SIGGRAPH '77), Vol. 11, No. 2, Summer 1977, pp. 65-71.

5) Catmull, Edwin, ''The Problems of Computer-Assisted Animation'' Computer Graphics (SIGGRAPH '78), Vol. 12, Num. 3, July '78, pp. 348-353.

6) Feiner, S., D. Salesin, T. Banchoff, ''Dial: A Diagrammatic Animation Language'', IEEE Computer Graphics and Applications, Vol. 2, Num. 7, Sept. '82, pp. 43-53.

7) Reynolds, Craig W., ''Computer Animation with Scripts and Actors,'' Computer Graphics, (Proc. SIGGRAPH '82), Vol. 16, Num. 3, July 1982, 289-296.

8) O'Donnell, T. J., Arthur J. Olson, ''GRAMPS - A Graphics Language Interpreter for Real-Time, Interactive, Three-Dimensional Picture Editing and Animation,'' Computer Graphics (Proc SIGGRAPH 1981), Vol. 15, Num. 3, Aug. 1981, pp. 133-142.

9) Magnenat-Thalmann, N., Daniel Thalmann, Mario Fortin, ''Miranim: An Extensible Director-Oriented System for the Animation of Realistic Images,'' IEEE Computer Graphics & Applications, Vol. 5, Num. 3, March 1985, pp. 61-73.

10) Thatch, B.R., A. Myklebust, ''A PHIGS-Based Graphics Input Interface for Spatial-Mechanism Design,'' IEEE Computer Graphics & Applications, Vol. 8, Num. 3, March 1988, pp. 26-38.

11) Giloth, Copper, Jane Veeder, ''The Paint Problem,'' IEEE Computer Graphics & Applications, Vol. 5, Num. 7, July 1985, pp. 66-75.

12) Magnenat-Thalmann, N., and D. Thalmann, ''Special Cinematographic Effects With Virtual Movie Cameras,'' IEEE Computer Graphics & Applications, Vol. 6, Num. 4, April 1986, pp 43-50.

13) Potmesil M., I. Chakravarty, ''Synthetic Image Generation with a Lens and Aperture Camera Model,'' ACM Trans. Graphics, Vol. 1, No. 2, April 1982, pp 85-108.

14) Magnenat-Thalmann, N., and D. Thalmann, ''Three-Dimensional Computer Animation: More an Evolution Than a Motion Problem,'' IEEE Computer Graphics & Applications, Vol. 5, Num. 10, October 1985, pp 47-57.

15) Foley, James D, Victor L. Wallace, Peggy Chan, ''The Human Factors of Computer Graphics Interaction Techniques.'' IEEE Computer Graphics & Applications, November 1984, Vol. 4, Num. 11, pp 13-48.

16) Csuri, C.; R. Hackathorn, R. Parent, W. Carlson and M. Howard, ''Towards an Interactive High Visual Complexity Animation System,'' Computer Graphics (Proc. SIGGRAPH 79), pp. 289-299.

17) Magnenat-Thalmann, Nadia and Daniel Thalmann, ''The Use of High-Level 3-D Graphical Types in the Mira Animation System,'' IEEE Computer Graphics and Applications, Vol. 3, num. 9, Dec. 1983, pp. 9-16.

18) Swezey, Robert W., Elaine G. Davis, ''A Case Study of Human Factors Guidelines in Computer Graphics,'' IEEE Computer Graphics and Applications, Vol. 3, Num. 8, Nov. 1983, pp. 21-30.

19) Gomez, Julian E., ''TWIXT: A 3D Animation System,'' Computers & Graphics, Vol. 9, Num. 3, 1985, pp. 291-298.

20) MaGuire, M. C., ''A Review of Human Factors Guidelines and Techniques for the Design of Graphical Human-Computer Interfaces,'' Computers & Graphics, Vol. 9, Num. 3, 1985, pp. 221-235.

21) Hearn, Donald, M. Pauline Baker, Computer Graphics, Prentice-Hall, Englewood Cliffs, NJ, 1986

22) Burtnyk, N., M. Wein, ''Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation,'' CACM, Oct. 1976

23) Stern, Garland, ''GAS - A System for Computer Aided Keyframe Animation,'' PHD dissertation, University of Utah, 1978.

24) Foley, James D., Charles F. McMath, ''Dynamic Process Visualization,'' IEEE Computer Graphics and Applications, Vol. 6, Num. 3, March 1986, pp. 16-25.

25) Shuey, David, David Bailey, Thomas P. Morrissey, ''PHIGS: A Standard, Dynamic, Interactive Graphics Interface,'' IEEE Computer Graphics and Applications, Vol. 6, Num. 8, August 1986, pp. 50-57.

Recommendations Towards a Direct Manipulation Interface for Three-Dimensional Computer Animation

by

CRAIG ARTHUR ORCUTT

B.S., Kansas State University, 1987

A THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Computer and Information Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

ABSTRACT

In recent years, several computer systems have been developed to aid the artist in producing three-dimensional animation. These systems have either become aids for the creation of, or the actual medium of the animation. Most of these systems, however, require the extensive use of complex director's languages which the animators are forced to learn and use. This is neither an intuitive nor cognitively bound technique to the activity to which animators are familiar. This thesis presents detailed recommendations towards the design of a direct manipulation interface for three-dimensional computer animation. The goal of this thesis is the direct transference of concepts and practices from the field of animation to the computer. These concepts include the instantiation of actor, photographic and photometric objects and the description of motion for those objects.